НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»

спеціальності 122 «Комп’ютерні науки та інформаційні технології»

на тему: «Система моніторингу рівня низьковуглецевого розвитку»

Виконав: студент IV курсу, групи ТМ-61

Анненков Максим Едуардович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доц., доц., к.е.н. Караєва Н.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020 р.

ЗАВДАННЯ

на дипломну роботу студенту

Анненкову Максиму Едуардовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Система моніторингу рівня низьковуглецевого розвитку України”
керівник роботи доц., к.е.н. Караєва Н.В.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____ _____ 2020 року
3. Вихідні дані до роботи ASP.NET Core проект
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Проаналізувати існуючи програмні рішення та засоби аналізу низьковуглецевого розвитку, спроектувати структуру бази даних спроектувати архітектуру програми, розробити програмне забезпечення, реалізувати інтерфейс користувача системи.
5. Перелік ілюстративного матеріалу:
Постановка задачі, вхідна та вихідна інформація, структура системи, діаграма прецедентів, засоби реалізації, архітектура програми, концептуальна схема БД, інтерфейс авторизації та головної сторінки, інтерфейс кореляційного аналізу то джерел даних, інтерфейс перегляду регіональної статистики, інтерфейс перегляду річної статистики, інтерфейс адміністрування бази даних, висновки.
7. Дата видачі завдання ”___”_____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	Захист програмного продукту		
8.	Передзахист		
9.	Захист		

Студент _____ Анненков М.Е.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Караєва Н.В.
(підпис) (прізвище та ініціали)

ВІДГУК

керівника дипломної роботи

освітньо-кваліфікаційного рівня „бакалавр”

виконаної на тему : “Система моніторингу рівня низьковуглецевого розвитку України”

студентом Анненковим Максимом Едуардовичем

Дипломна робота Анненкова М.Е. присвячена розробці системи моніторингу рівня низьковуглецевого розвитку України. Тема дипломної має практичну спрямованість у сфері екологічного та економічного розвитку і є актуальною. Створення системи моніторингу, що є загальнодоступним інформаційним ресурсом має сприяти демократичним перетворенням у суспільстві.

У ході виконання роботи студентом було проведено аналіз предметної області та визначено завдання, які мають вирішуватись розробленою системою. Автор роботи вміє аналізувати літературні джерела, самостійно розробляти програмне забезпечення згідно з поставленою задачею.

Студент проявив високу обізнаність у прийнятті сучасних рішень, уміння аналізувати статистичну інформацію, приймати правильні технологічні рішення. Пояснювальна записка та документація оформлена належним чином у відповідності до прийнятих стандартів.

Також студент Анненков М. Е. прийняв участь у XVIII Міжнародній науково-практичній конференції аспірантів, магістрантів, студентів «Сучасні проблеми наукового забезпечення енергетики».

Зважаючи на вищезазначене, студент Анненков М.Е. заслуговує на присвоєння кваліфікації бакалавра комп’ютерних наук та інформаційних технологій з напряму підготовки 122 “Комп’ютерні науки та інформаційні технології” за спеціалізацією “Інформаційні технології моніторингу довкілля”.

Керівник дипломної роботи

доц., доц., к.е.н. _____ Караєва Н.В.

(посада, вчені звання, ступінь) (підпис) (ініціали, прізвище)

РЕЦЕНЗІЯ

на дипломну роботу

освітньо-кваліфікаційного рівня „бакалавр”

виконаної на тему : “Система моніторингу рівня низьковуглецевого розвитку України”

студентом Анненковим Максимом Едуардовичем

Представлена до рецензії бакалаврська робота відповідає затвердженій темі, а також є актуальною на даний, оскільки моніторинг еколого-економічних показників є важливим інструментом політики забезпечення, як низьковуглецевого так і еколого-економічного розвитку України.

Пояснювальна записка містить опис результатів виконання дипломної роботи, відповідає затвердженій темі та виконана відповідно до завдання. Також у пояснювальній записці наявний опис архітектури програмних модулів, приклади роботи користувача с програмним додатком, приклади побудови діаграм та графіків.

Анненков М.Е. у своїй роботі застосував мови програмування C# та JavaScript, фреймворк ASP.NET Core з використанням патерну MVC та Entity Framework Core, HTML5, CSS.

Робота має чітку структуру. Оформлення документації виконано на високому рівні. Робота містить додатки і ілюстрації.

Дипломна робота виконана у повному обсязі та заслуговує оцінки “відмінно”, а студент Анненков М. Е. заслуговує на присвоєння кваліфікації бакалавра комп’ютерних наук та інформаційних технологій з напряму підготовки 122 “Комп’ютерні науки та інформаційні технології” за спеціалізацією “Інформаційні технології моніторингу довкілля”.

Рецензент

_____	_____	_____
(посада, вчені звання, ступінь)	(підпис)	(ініціали, прізвище)

АНОТАЦІЯ

Метою дипломної роботи була розробка системи моніторингу низьковуглецевого розвитку України, що є загальнодоступним інформаційним ресурсом. Система дозволяє, на основі статистичних даних, вимірювати рівень низьковуглецевого розвитку та проводити моніторинг його динаміки. Це дасть можливість зважено та чітко проаналізувати ефективність роботи діючих заходів на зменшення обсягу викидів парникових газів, крім того, вплив цих заходів на подальше економічне зростання.

Записка складається з 5 розділів, 50 сторінок, 25 рисунків, 5 таблиць та 10 використаних джерел

ABSTRACT

The purpose of the thesis was to develop a system for monitoring low-carbon development of Ukraine, which is a publicly available information resource. Which will allow, on the basis of statistical data, to measure the level of low carbon development and monitor its dynamics. This will provide an opportunity to carefully and clearly analyze the effectiveness of existing measures to reduce greenhouse gas emissions, in addition, the impact of these measures on further economic growth.

The note consists of 5 sections, 50 pages, 25 figures, 5 tables and 10 sources used

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СКБД — система керування базами даних.

API (англ. Application Programming Interface) — прикладний програмний інтерфейс.

БД — база даних.

CRUD (англ. create read update delete) — 4 базові функції управління даними «створення, зчитування, зміна і видалення».

ORM — англ. Object-relational mapping, Об'єктно-реляційна проекція) — технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування.

SQL (англ. Select query language) — декларативна мова програмування

для взаємодії користувача з базами даних.

MS (англ. Microsoft) — багатонаціональна корпорація комп'ютерних технологій.

HTML (англ. HyperText Markup Language — мова розмітки гіпертексту) — це мова тегів, якою пишуться гіпертекстові документи для мережі Інтернет;

CSS (англ. Cascading Style Sheets, укр. Каскадні таблиці стилів) — спеціальна мова стилю сторінок.

НВР — низьковуглецевий розвиток.

ПГ — парниковий газ.

ВВП — валовий внутрішній продукт

ВРП —валовий регіональний продукт.

ВСТУП

18 липня 2018 року на засіданні Кабінету Міністрів України членами Уряду було прийнято стратегічний документ переходу економіки України на модель низьковуглецевого розвитку (НВР). Як зазначено в «Стратегії низьковуглецевого розвитку України до 2050 року» (подалі Стратегія), низьковуглецевий розвиток – це розвиток щодо відокремлення подальшого економічного зростання та соціального розвитку держави від збільшення обсягу викидів парникових газів [1].

Для вимірювання рівня НВР, моніторингу його динаміки розробляється і використовується багатопараметрична система відповідних статистичних показників та індикаторів. Створення системи моніторингу, що є загальнодоступним інформаційним ресурсом має сприяти демократичним перетворенням у суспільстві шляхом забезпечення доступу населення до інформаційних ресурсів.

Робота містить 5 розділів.

У першому розділі описується постановка задачі створення багатопараметричної системи моніторингу низьковуглецевого розвитку України, що є загальнодоступним інформаційним ресурсом.

У другому розділі описуються підходи до вирішення проблеми створення багатопараметричної системи моніторингу низьковуглецевого розвитку України.

У третьому розділі вказуються основні засоби розробки даної системи.

У четвертому розділі дано опис реалізованого програмного продукту та його архітектури.

У п’ятому розділі описано роботу користувача з програмною системою.

1. ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ НИЗЬКОВУГЛЕЦЕВОГО РОЗВИТКУ УКРАЇНИ

Призначення програмної системи моніторингу полягає в оцінюванні динаміки загроз прийнятного рівня низьковуглецевого розвитку (НВР) регіонів України для подальшого реагування і вироблення регіональної політики, щодо їх мінімізації. Для моніторингу використовується система відповідних статистичних показників та індикаторів.

Створення системи моніторингу НВР, що є загальнодоступним інформаційним ресурсом, має сприяти демократичним перетворенням у суспільстві шляхом забезпечення доступу населення до інформаційних ресурсів. Основні блоки системи зображено на рисунку 1.1.

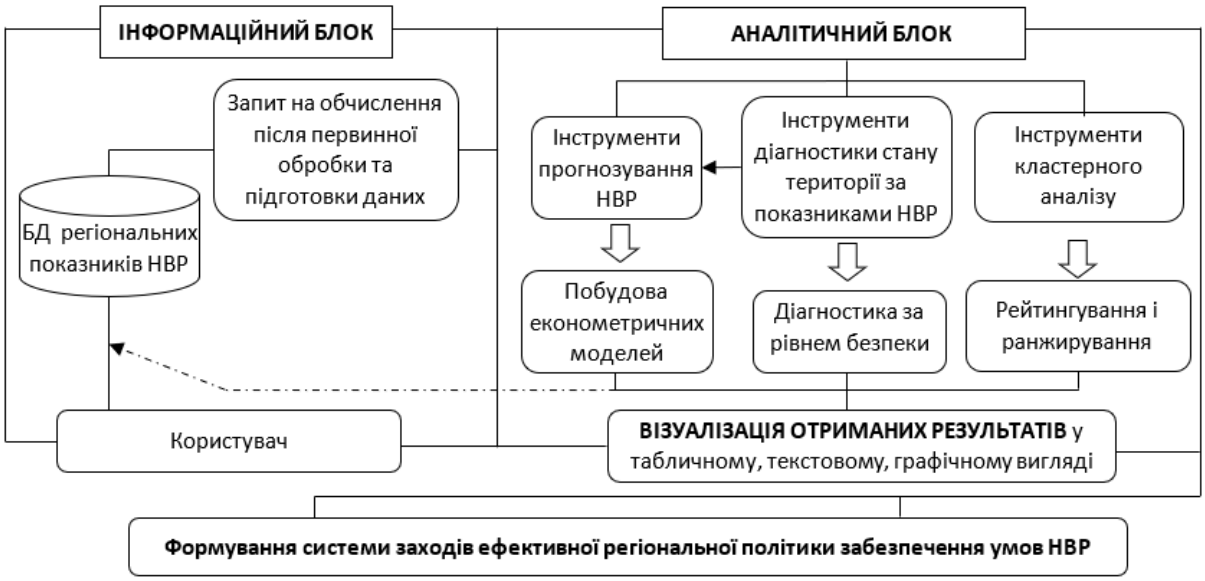


Рис. 1.1– Структура системи моніторингу показників НВР

В результаті розробки програмного продукту, користувачу буде надана можливість опрацьовувати дані різних показників та індикаторів, усіх регіонів за роками, прогнозувати варіанти подальшого НВР і відбирати найбільш ефективні стратегії відокремлення подальшого економічного зростання та

соціального розвитку держави від збільшення обсягу викидів парникових газів.

Розроблена система повинна забезпечувати наступні можливості:

– аналіз динаміки зміни викидів парникових газів НВР;
– визначення ступені кореляційних зв’язків між показниками НВР;
– рейтингування регіонів за рівнем викидів парникових газів;
– візуалізація отриманих результатів у табличному, текстовому та графічному вигляді для подальшого аналізу.
– зберігати звіти на особистий комп’ютер;
– можливість доповнювати базу даних (БД).

Вхідна інформація – статистичні показники з сайту Державної служби статистики України.

Для розробки системи була використана мова програмування C#. Розробка графічного інтерфейсу користувача відбувалась на основі JavaScript, HTML5 та CSS.

Метою розробки системи моніторингу низьковуглецевого розвитку України є створення програмного продукту, що дасть змогу, на основі статистичних даних, вимірювати рівень НВР та займатись моніторингом його динаміки. Це дасть можливість зважено та чітко проаналізувати ефективність роботи діючих заходів на зменшення обсягу викидів парникових газів, крім того, вплив цих заходів на подальше економічне зростання. Адже стратегія НВР України, з одного боку, спирається на національні пріоритети сталого розвитку та чинної стратегії розвитку секторів економіки, а з іншого боку, визначає можливу траєкторію економічного зростання з урахуванням цілей державної політики зі скорочення викидів і збільшення поглинання ПГ.

Мета цього проекту полягає в тому, щоб дати змогу звичайному користувачу ознайомитись з рівнем та відстежувати динаміку рівня НВР, також допомогти у визначенні стратегічних напрямів переходу економіки України на траєкторію низьковуглецевого зростання на засадах сталого розвитку відповідно до національних пріоритетів.

В залежності від ролі користувача програма надає різні функціональні можливості.

В випадку, якщо вхід в програму був здійснений під користувачем з роллю Адміністратор, то інтерфейс користувача дозволяє:

– продивитися і відредагувати базу даних статистичних показників;

– актуалізувати дані за нові періоди;

– видаляти дані з таблиць БД.

У випадку коли вхід в програму був здійснений під користувачем з роллю Аналітик, то у користувача є такі можливості:

– аналіз динаміки показників НВР;

– перегляд регіональної статистики;

– перегляд річної статистики;

– кореляційний аналіз.

2. ОГЛЯД ІСНУЮЧИХ МЕТОДІВ АНАЛІЗУ НИЗЬКОВУГЛЕЦЕВОГО РОЗВИТКУ

У розділі обґрунтовано нормативно-методичну основу формування системи індикаторів НВР України. Визначено необхідність використання методів кореляційного аналізу й рейтингування як основних елементів аналітичного блоку системи моніторингу НВР.

2.1 Основи формування системи індикаторів низьковуглецевого розвитку

Як зазначається в багатьох міжнародних, вітчизняних доповідей і наукових видань, глобальна зміна клімату стає однією з найгостріших екологічних проблем, які стоять перед людством. Саме тому важливим напрямом досягнення цілей сталого розвитку є усунення негативних наслідків зміни клімату [3] (рисунок 2.1).

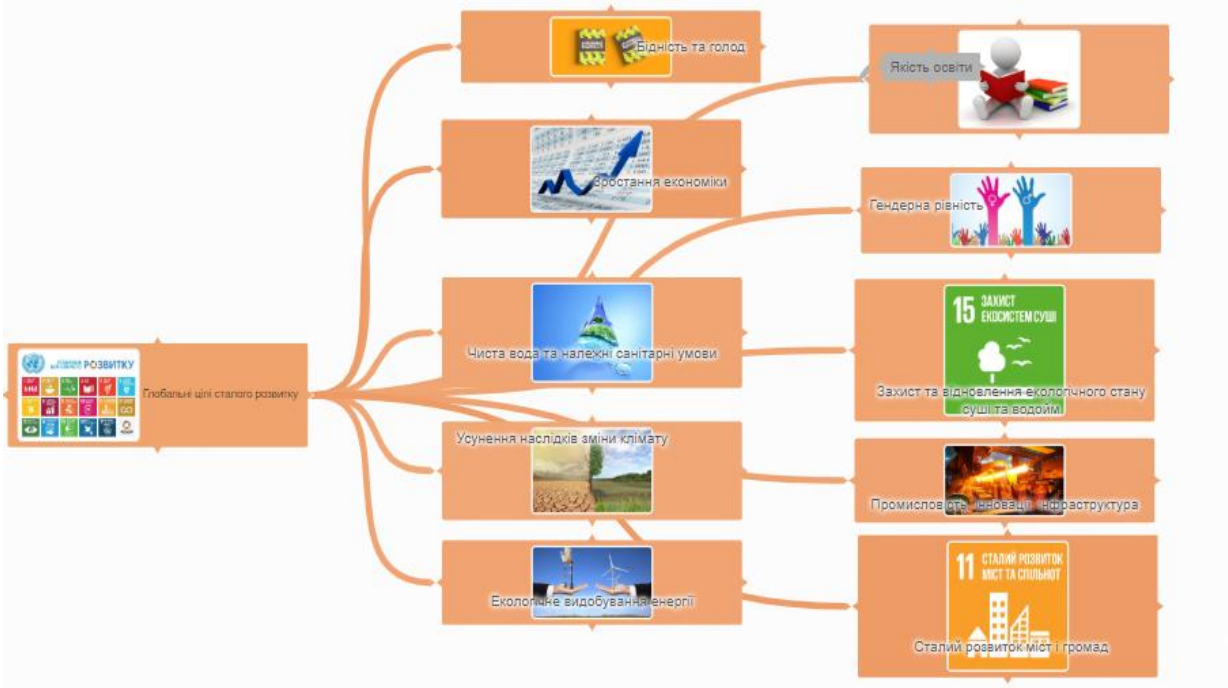


Рисунок 2.1 – Глобальні цілі сталого розвитку

Новітні наукові дані свідчать, що глобальні кліматичні зміни можуть привести до таких наслідків як: зростання кількості природних катаклізмів;

утворення непридатних для життя територій; скорочення біологічної різноманітності планети; нестача питної води, голод і епідемії; підвищення рівня світового океану та ін.

У багатьох наукових дослідженнях доведено, що вплив антропогенної виробничої діяльності на глобальний клімат пов'язаний, в першу чергу, із збільшення кількості парникових газів (вуглекислого газу, метану, закису азоту, хлорфторвуглецiв та ін.) , що посилює парниковий ефект в атмосфері. Також ризик-чинником появі негативних наслідків зміни клімату є низький рівень екологізації економіки в слабо розвинутих країн. Виходячи з вищезазначених положень, екологізація економіки і перехід до низьковуглецевого розвитку є основними механізмами досягнення цілей сталого розвитку (рисунок 2.2).

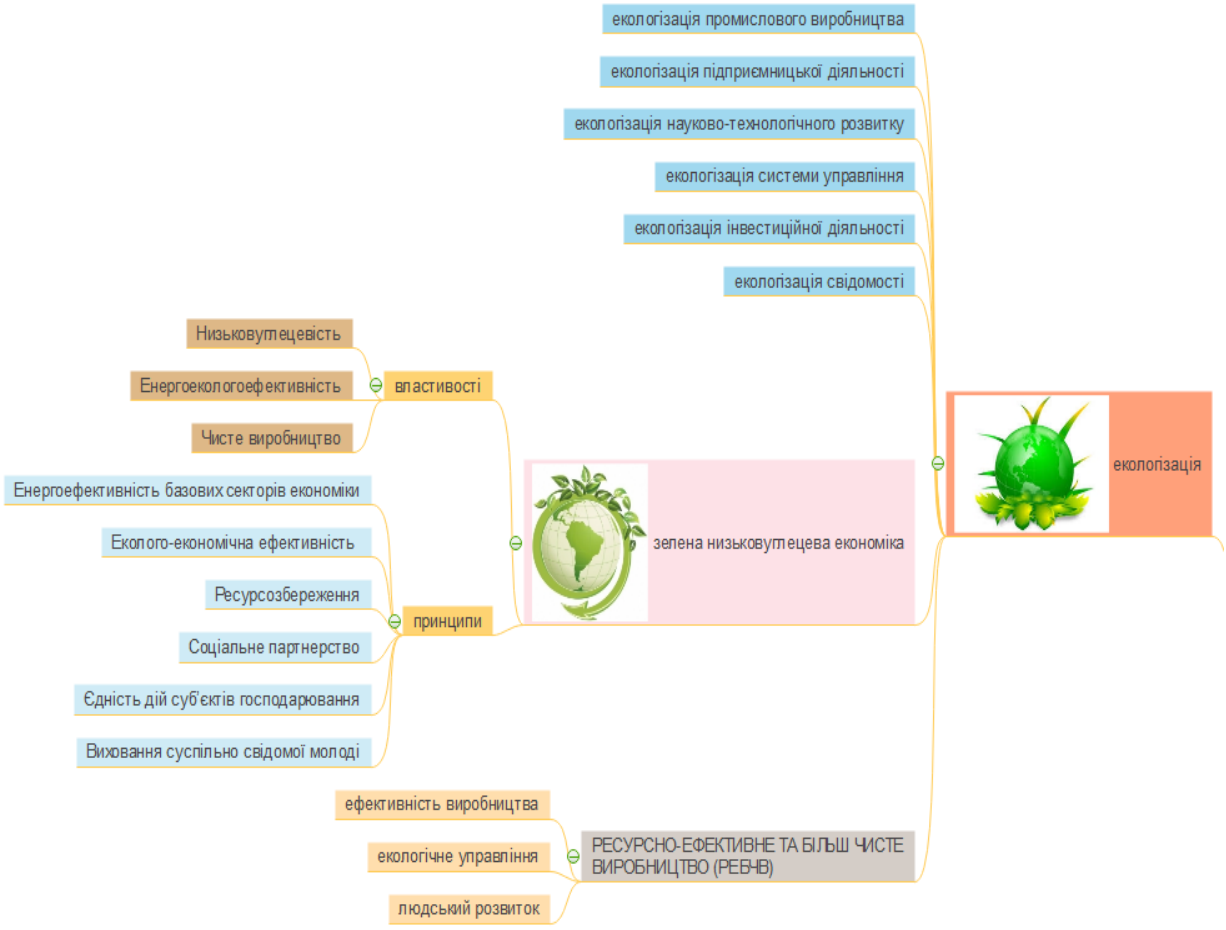


Рисунок 2.2 – Основні механізми досягнення цілей сталого розвитку

Виходячи з основних положень Стратегії, основним принципом НВР має бути зростання економіки за умови скорочення викидів парникових газів, а його

критеріями – скорочення вуглецевих викидів, зростання валового внутрішнього продукту (ВВП).

Для вимірювання НВР, моніторингу його динаміки пропонуються використовувати багатопараметричну систему відповідних статистичних показників, зокрема:

Індикатори антропогенного впливу на рівень НВР:

- валовий регіональний продукт (ВРП)(млн. грн);
- кількість великих підприємств;
- утворення відходів (тис. т.);
- витрати на охорону довкілля (млн грн).

Показники забруднення атмосферного повітря:

- обсяг викидів забруднюючих речовин – усього (тис. т);
- викиди оксиду вуглецю, (тис. т.);
- викиди метану, (тис. т.);
- викиди діоксиду азоту, (тис. т.);
- викиди оксиду азоту, (тис. т.);
- викиди сажі, (тис. т.);
- викиди діоксиду сірки, (тис. т.);
- викиди неметанових легких органічних сполук, (тис. т.);
- викиди діоксиду вуглецю, (млн. т.);
- викиди від пересувних джерел забруднення (тон).

2.2 Методи аналізу

Викиди парникових газів в атмосферу залежать від обсягів виробництва підприємств еколого-небезпечніших галузей промисловості. Тобто мова йде про необхідність виявлення причинно-наслідкових (кореляційних) зав’язків НВР. Кореляція (від лат. correlatio – співвідношення) – це статистична залежність між випадковими величинами, що носить імовірнісний характер. Суть причинного зв'язку полягає в тому, що при необхідних умовах одне

явище зумовлює інше і в результаті такої взаємодії виникає наслідок. Кореляційна залежність виникає тоді, коли одна з величин залежить не тільки від заданої другої, а й від деяких випадкових факторів; або, коли серед умов, від яких залежать обидві величини, є загальні для них обох [4].

Найпростішим видом кореляційного зв'язку є зв'язок між двома ознаками: результативною і факторною. Саме тому важливим методичним інструментом аналітичного блоку системи моніторингу НВР України є кореляційний аналіз.

В задачах моніторингу НВР необхідно виявити залежність між двома властивостями (ознаками) *x* і *y* одного і того ж економічного об'єкту, або між певними ознаками різних об’єктів. Якщо вказані ознаки допускають кількісне вимірювання, і, виходячи з економічної або екологічної характеристики об'єкту, ознака *y* залежить від ознаки *x*, тоді *x* можна назвати незалежною змінною, або факторною ознакою, або просто фактором, а *y* – залежною змінною або результативною ознакою.

Якщо кожному значенню факторної ознаки *x* відповідає одне і тільки одне значення результативної ознаки *y*, то говорять, що між цими ознаками існує функціональний зв’язок: *y = f(x)*.

В економічних дослідженнях взаємозв'язку двох факторів серед множини функцій часто розглядається прямолінійна форма зв'язку, яка виражається рівнянням прямої лінії. Взаємозв’язок між окремими ознаками описується за допомогою коваріаційної або кореляційної матриці [4].

Кореляційна залежність проявляється тільки у масових явищах і може встановлюватися для пари показників (однофакторна кореляція) або для декількох показників (багатофакторна кореляція).

Одним з основних показників щільності кореляційного зв'язку показника *y* зі всіма факторами $X_i\ (i=1,2,...,m)$, а також показника ступеня близькості математичної форми зв'язку до вибіркових даних є коефіцієнт багатофакторної кореляції, який має вигляд

$$R = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \hat{\bar{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2 \sum_{i=1}^{n}(\hat{y}_i - \hat{\bar{y}})^2}}.$$

Квадрат коефіцієнта багатофакторної кореляції називається коефіцієнтом детермінації і позначається через R^2,

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y - \bar{y})^2}.$$

Коефіцієнт детермінації показує, скількома відсотками варіація залежної змінної визначається варіацією незалежних змінних.

Коефіцієнт кореляції R_{xy} – це відносна міра зв'язку між двома факторами. Тому значення коефіцієнта кореляції завжди розташовані між -1 та +1, тобто $-1 < R_{xy} < 1$.

Додатне значення коефіцієнта кореляції свідчить про прямий зв'язок між показниками, а від'ємне – про зворотній зв'язок.

Якісне оцінювання ступеня зв’язку випадкових величин може виконуватися з використанням коефіцієнта кореляції за шкалою Чеддока (таблиця 2.1).

Таблиця 2.1. Шкала Чеддока

Значення коефіцієнта кореляції	Зв’язок
[0,1...0,3)	незначний
[0,3...0,5)	помірний
[0,5...0,7)	істотний
[0,7...0,9)	високий
[0,9...0,99]	дуже високий
1,0	функціональний

3. ЗАСОБИ РОЗРОБКИ

Важливим чинником, під час розробки програмного продукту, є вибір засобів програмної реалізації та технологій. Проаналізувавши поставлену задачу та методи її вирішення, буле вирішено розробити програмний продукт у вигляді веб-додатку.

3.1 Архітектура програмного продукту

Для розробки програмного продукту у вигляді веб-додатку, який буде працювати зі статистичною інформацією, було обрано наступну архітектуру: база даних – сервер – клієнт (рисунок 3.1)

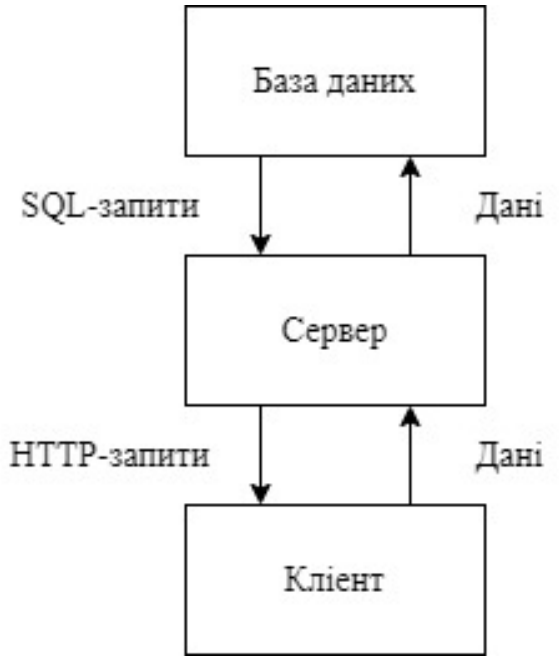


Рисунок 3.1 – Архітектура програм

Під час експлуатації програми, користувач взаємодіє з клієнтським додатком за допомогою веб-інтерфейсу. Запити до серверу надходять у вигляді HTTP-запитів.

На сервері відбувається ідентифікація ролі користувача, для надання необхідних додаткових прав доступу до налаштувань та редагування бази даних. Для уникнення можливості пошкодження даних, доступ до бази даних можливий тільки через сервер.

База даних забезпечує надійне збереження даних, забезпечує цілістність даних за допомогою зовнішніх ключів та зв’язків.

3.2 Шаблон проектування серверу

Шаблон проектування — це архітектура, рішення яке описує яким чином вирішуються задачі, які часто зустрічаються при розробці програмних систем чи додатків.

Для реалізації серверу було використано фреймворк, який реалізує шаблону проектування MVC.

Модель–представлення–контролер (Model-View-Controller ,MVC) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Шаблон MVC – це архітектурний шаблон, що часто використовується для розробки призначених для користувача інтерфейсів, який розділяє додаток на три частини, щоб відокремити внутрішнє представлення інформації від того, як інформація може надаватися і приймається від користувача. Шаблон проектування MVC розділяє ці основні компоненти, дозволяючи повторне використання коду і паралельну розробку.

Концепція шаблону MVC передбачає поділ додатка на три компоненти:

– **Модель** (model): описує використовувані в додатку дані, а також логіку, яка пов'язана безпосередньо з даними, наприклад, логіку валідації даних. Як правило, об'єкти моделей зберігаються в базі даних.

  У MVC моделі представлені двома основними типами: моделі уявлень, які використовуються уявленнями для відображення і передачі даних, і моделі домену, які описують логіку управління даними.

  Модель може містити дані, зберігати логіку управління цими даними. У той же час модель не повинна містити логіку взаємодії з

користувачем і не має визначати механізм обробки запиту. Крім того, модель не повинна містити логіку відображення даних в поданні.

– **Представлення** (view): відповідають за візуальну частину або призначений для користувача інтерфейс, нерідко html-сторінка, через який користувач взаємодіє з додатком. Також уявлення може містити логіку, пов'язану з відображенням даних. У той же час уявлення не повинно містити логіку обробки запиту користувача або управління даними.
– **Контролер** (controller): представляє центральний компонент MVC, який забезпечує зв'язок між користувачем та програмою, поданням і сховищем даних. Він містить логіку обробки запиту користувача. Контролер отримує вводяться користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання, наповненого даними моделей [7].

Відносини між компонентами шаблону можна описати наступною схемою (рисунок 3.2).

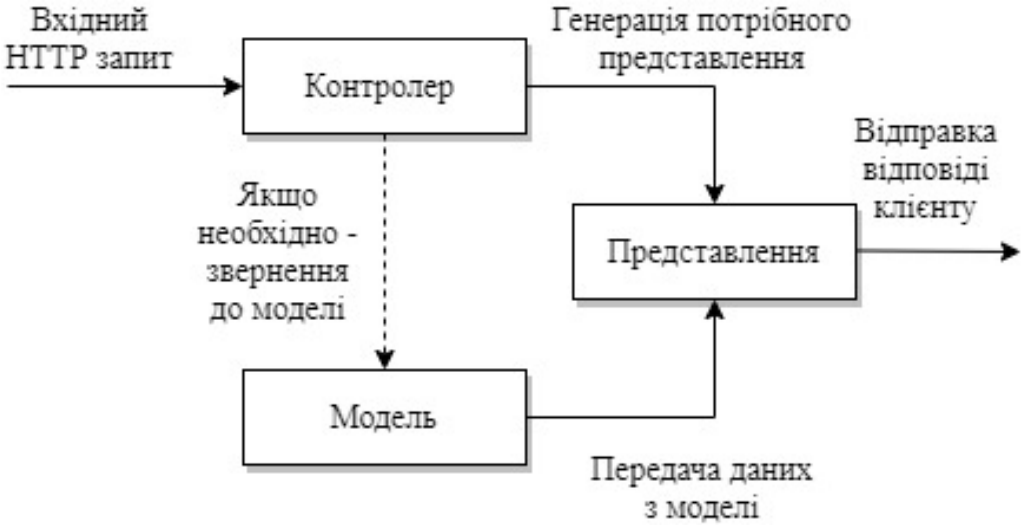


Рисунок 3.2 – Схема роботи шаблону MVC

Головними цілями MVC э створення умов для паралельної розробки та можливості повторного використання коду. У цілому MVC поділяє різні

компоненти додатки, розробники можуть працювати паралельно над різними компонентами, не впливаючи і не блокуючи іншу.

Таке розмежування компонентів додатка дозволяє реалізувати концепцію розділення відповідальності, при якій кожен компонент відповідає за свою строго окреслену сферу. У зв'язку з чим легше побудувати роботу над окремими компонентами. І завдяки цьому додаток легше розробляти, підтримувати і тестувати окремі компоненти. Припустимо, якщо нам важлива візуальна частина або фронтенд, то ми можемо тестувати представлення незалежно від контролера. Або ми можемо зосередитися на бекенда і тестувати контролер.

Також, такий шаблон має свої недоліки, які можуть викликати накладні витрати на неправильно розроблене програмне забезпечення, наприклад:

– необхідність бути кваліфікованим одразу в декількох технологіях;
– складна адаптація до критеріїв декомпозиції, складна навігація фреймворків, через запровадження нових рівней абстракції,
– додаткові вимоги від розробників збереження узгодженості декількох частин одночасно, для запобігання розсіюванню, через розкладання функції на три артефакти.

3.3 Платформа

Платформа ASP.NET Core – технологія, розроблена компанією Microsoft, яка була призначена для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів.

З одного боку, ASP.NET Core – це продовження розвитку платформи ASP.NET. Але з іншого боку, це не просто черговий реліз. Вихід ASP.NET Core означає революцію всієї платформи.

ASP.NET Core тепер повністю є відкритим фреймворком. Всі початкові файли фреймворку доступні на GitHub. Що є великим плюсом для

розробників, бо при виникненні запитань щодо роботи та компонентів платформи, можна звернутися до фалу початкового коду.

ASP.NET Core може працювати поверх крос-платформенного середовища .NET Core, яка може бути розгорнута на основних популярних операційних системах: Windows, Mac OS, Linux. І таким чином, за допомогою ASP.NET Core ми можемо створювати крос-платформні додатки. І хоча Windows як середовище для розробки і розгортання програми досі превалює, але тепер вже ми не обмежені тільки цією операційною системою. Тобто ми можемо запускати веб-додатки не тільки на ОС Windows, але і на Linux і Mac OS. А для розгортання веб-додатки можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

Одним з характерних моментів платформи ASP.NET Core є застосування шаблону MVC. Причому остання версія MVC-фреймворка, який застосовується в ASP.NET Core, має номер 3.0 / 3.1. Тому важливо не плутати ASP.NET MVC 5, який застосовується в ASP.NET 4.5-4.8, і фреймворк MVC, який застосовується в ASP.NET Core. Хоча в багатьох аспектах ці фреймворки будуть збігатися [7].

Завдяки модульності фреймворка всі необхідні компоненти веб-додатки можуть завантажуватися як окремі модулі через пакетний менеджер Nuget. Крім того, на відміну від попередніх версій платформи немає необхідності використовувати бібліотеку System.Web.dll.

ASP.NET Core включає в себе фреймворк MVC, який об'єднує функціональність MVC, Web API і Web Pages. У попередніх версії платформи дані технології реалізувалися окремо і тому містили багато дублюючої функціональності. Зараз же вони об'єднані в одну програмну модель ASP.NET Core MVC. А Web Forms повністю пішли в минуле.

Також невірно ототожнювати ASP.NET Core цілком з фреймворком ASP.NET Core MVC. Фреймворк ASP.NET Core MVC працює поверх платформи ASP.NET Core, і призначений для того, щоб спростити створення

програми. Але ми можемо і не використовувати MVC, а застосовувати чистий ASP.NET Core і на ньому цілком вибудовувати логіку програми.

Сам шаблон MVC не є якоюсь новою ідеєю в архітектурі додатків, він з'явився ще в кінці 1970-х років в компанії Xerox як спосіб організації компонентів в графічному додаток на мові Smalltalk [7].

Крім об'єднання вищезазначених технологій в одну модель MVC був доданий ряд додаткових функцій.

Однією з таких функцій є тег-хелпери (tag helper), які дозволяють більш органічно поєднувати синтаксис html з кодом C #.

ASP.NET Core характеризується розширюваністю. Фреймворк побудований з набору щодо незалежних компонентів. І ми можемо або використовувати вбудовану реалізацію цих компонентів, або розширити їх за допомогою механізму спадкування, або зовсім створити і застосовувати свої компоненти зі своїм функціоналом.

Для обробки запитів тепер використовується новий конвеєр HTTP, який заснований на компонентах Katana і специфікації OWIN. А його модульність дозволяє легко додати свої власні компоненти.

Якщо підсумувати, то можна виділити наступні ключові відмінності ASP.NET Core:

- новий легкий і модульний конвеєр HTTP-запитів;
- можливість розгортати додаток як на IIS, так і в рамках свого власного процесу;
- використання платформи .NET Core і її функціональності;
- поширення пакетів платформи через NuGet;
- інтегрована підтримка для створення та використання пакетів NuGet;
- єдиний стек веб-розробки, що поєднує Web UI і Web API;
- конфігурація для спрощеного використання в хмарі;
- вбудована підтримка для впровадження залежностей;
- можливість розширення;

– кросплатформеність: можливість розробки і розгортання додатків ASP.NET на Windows, Mac і Linux;
– розвиток як open source, відкритість до змін.

Ці та інші особливості і можливості стали основою для нової моделі програмування.

3.4 Середовище розробки

Microsoft Visual Studio 2019 повністю підтримує створення веб-додатків за допомогою ASP.NET Core, ASP.NET (.NET Framework), HTML, JavaScript і контейнерів, включаючи підтримку Docker.

Обране середовище дозволяє підвищити ефективність розробки веб-додатків .NET за допомогою ASP.NET Core технологій на основі таких стандартів, як HTML і JavaScript.

Застосування в розробці:

– веб-сайт, який використовує Razor Pages в ASP.NET Core;

– веб-API з ASP.NET Core MVC;

– веб-додатки реального часу на основі ASP.NET Core SignalR компоненти;

– засоби розробки для .NET Framework 4.x;

– засоби розробки .NET Core 2.1;

– ASP.NET і засоби веб-розробки;

– засоби профілювання .NET;

– засоби розробки контейнерів;

– хмарні кошти для веб-розробки.

3.5 Клієнтська частина

На платформі ASP.NET Core для формування клієнтського застосунку у вигляді веб-інтерфейсу використовується layout view.

Веб-інтерфейс – це сукупність засобів, за допомогою яких користувач може взаємодіяти з веб-застосунком або веб-сайтом через браузер. Веб-інтерфейси отримали широке поширення у зв'язку із зростанням популярності всесвітньої павутини і відповідно повсюдного розповсюдження веб-браузерів.

Однією з основних вимог до веб-інтерфейсу є їх однаковий зовнішній вигляд і однакова функціональність при роботі в різних браузерах.

Класичним і найпопулярнішим методом створення веб-інтерфейсів є використання HTML із застосуванням CSS, для створення макету та його стиліцзації. У поєднанні з JavaScript, для динаміки веб-додатку, як правило за допомогою скриптових мов на стороні сервера та бібліотеки Boostrap для створення графічних інтерфейсів.

Бібліотека Bootstrap – це набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

В нашому випадку зручно буде працювати з веб інтерфейсом за допомогою тег-хелперів (tag helper) та Razor для сервеного рендерингу html коду, які дозволяють більш органічно поєднувати синтаксис html з кодом C #.

Інструмент Razor – це синтаксис розмітки, який дозволяє вставляти серверний код у веб-сторінки за допомогою C #. Це не мова програмування. Це мова розмітки на стороні сервера. Мова, яку Razor використовує для розмітки, за замовчуванням - це HTML. Візуалізація HTML з розмітки Razor не відрізняється від рендеринга HTML з файлу HTML. Razor використовує символ @ для переходу з HTML на C#. Razor аналізує вирази C # і передає їх у вихідний HTML.

У якості мови програмування, було обрано, C# та платформа .NET Core, фреймворк для створення RESTful сервісів ASP.NET Core MVC, ORM для доступу до бази даних Entity Framework Core.

Мова C # - це об'єктно-орієнтована мова програмування. Розроблена в 1998–2001 роках групою інженерів Microsoft, під керівництвом Андерса Хейлсберга і Скотта Вілтомута , як мова для розробки додатків для Microsoft .NET Framework. Згодом він був стандартизований як ECMA-334 і ISO / IEC 23270.

Мова C # відноситься до сімейства мов з C-подібним синтаксисом, його синтаксис найбільш близький до C ++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження оператора (включаючи явні та неявні перетворення типу), делегатів, атрибутів, подій, властивостей, загальних типів і методів, ітераторів, анонімних функцій з закриттями, LINQ, виключень, коментарів у форматі XML.

Зайнявши чимало від своїх попередників - C ++, Pascal, Modula, Smalltalk і, зокрема, Java - C #, спираючись на практику їх використання, виключає деякі моделі, які виявилися проблематичними при розробці програмних систем. Наприклад, C #, на відміну від C ++ і деяких інших мов, не підтримує успадкування декількох класів (тим часом дозволено успадкування декількох інтерфейсів) [10].

3.6 Взаємодія з базою даних

В якості системи керування базою даних (СКБД) було обрано Microsoft SQL Server. MS SQL Server є однією з найбільш популярних систем управління базами даних (СКБД) в світі. Дана СУБД підходить для самих різних проектів: від невеликих додатків до великих високонавантажених проектів.
SQL Server довгий час був винятково системою управління базами даних для Windows, проте починаючи з версії 16 ця система доступна і на Linux.

SQL Server характеризується такими особливостями як:

- продуктивність. SQL Server працює дуже швидко;
- надійність і безпека. SQL Server надає шифрування даних;

– простота. З даної СКБД відносно легко працювати і вести адміністрування [11].

ORM для доступу до бази даних - Entity Framework Core.

Entity Framework Core (EF Core) являє собою об'єктно-орієнтовану технологію від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом (object-relational mapping - відображення даних на реальні об'єкти).

Тобто EF Core дозволяє працювати базами даних, але є більш високий рівень абстракції: EF Core дозволяє абстрагуватися від самої бази даних і її таблиць і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework Core, ми вже працюємо з об'єктами.

Entity Framework Core підтримує безліч різних систем баз даних. Таким чином, ми можемо через EF Core працювати з будь-якої СУБД, якщо для неї є потрібний провайдер.

За замовчуванням на даний момент Microsoft надає ряд вбудованих провайдерів, зокрема для роботи з:

– MS SQL Server;
– SQLite;
– PostgreSQL;
– також є провайдери від сторонніх постачальників, наприклад, для MySQL.

Також варто відзначити, що EF Core надає універсальний API для роботи з даними. І якщо, наприклад, ми вирішимо змінити цільову СУБД, то основні зміни в проекті будуть стосуватися насамперед конфігурації і настройки підключення до відповідних провайдерам. А код, який безпосередньо працює з даними, отримує дані, додає їх в БД тощо, залишиться незміненим [12].

Entity Framework Core багато успадкував від своїх попередників, зокрема, Entity Framework 6. У той же час треба розуміти, що EF Core – це не

нова версія по відношенню до EF 6, а зовсім інша технологія, хоча в цілому принципи роботи у них будуть збігатися. Тому в рамках EF Core використовується своя система версій. Поточна версія - 3.0 була випущена у вересні 2019 року. І технологія продовжує розвиватися.

Як технологія доступу до даних Entity Framework Core може використовуватися на різних платформах стека .NET. Це і стандартні платформи типу Windows Forms, консольні додатки, WPF, UWP і ASP.NET Core. При цьому кроссплатформенная природа EF Core дозволяє задіяти її на таких платформах, як:

- ОС Windows;
- Linux;
- Mac OS X.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність визначає набір даних, які пов'язані з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх колекціями.

Відмінною рисою Entity Framework Core, як технології ORM, є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо створювати різні запити на вибірку об'єктів, в тому числі пов'язаних різними асоціативними зв'язками. А Entity Framework при виконання запиту транслює вираження LINQ в вирази, зрозумілі для конкретної СУБД (як правило, в вирази SQL).

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Застосунок буде представлений у вигляді веб-додатку. Перше, що повинен зробити користувач – авторизуватися. Авторизувавшись, як користувач, він потрапить на головну сторінку, де може ознайомитись з основними принципами стратегії НВР, отримати посилання на першоджерела стратегії, переглянути список показників, які використовує програма для моніторингу. Головне меню буде доступно з будь-якого місця програми та буде розташоване у верхній частині сторінки. Такий підхід зробить інтерфейс інтуїтивно зрозумілим та простим. Використовуючи головне меню, користувач зможе перейти до регіональної статистики та переглянути дані окремо обраного регіону за усі періоди як в табличному, так і в графічному вигляді, або перейти до статистики за роками і побачити дані усіх регіонів за окремий рік. Також програма повинна мати опис та посилання на першоджерела джерела кожного з показників. Крім того, одним із пунктів головного меню буде «Кореляція», де користувач зможе ознайомитись з кореляційною матрицею. Авторизувавшись, у ролі адміністратор, користувач отримає додаткові права на перегляд, редагування та актуалізацію бази даних.

4.1 Опис функціональності системи

В залежності від ролі користувача програма надає різні функціональні можливості (рисунок 4.1).

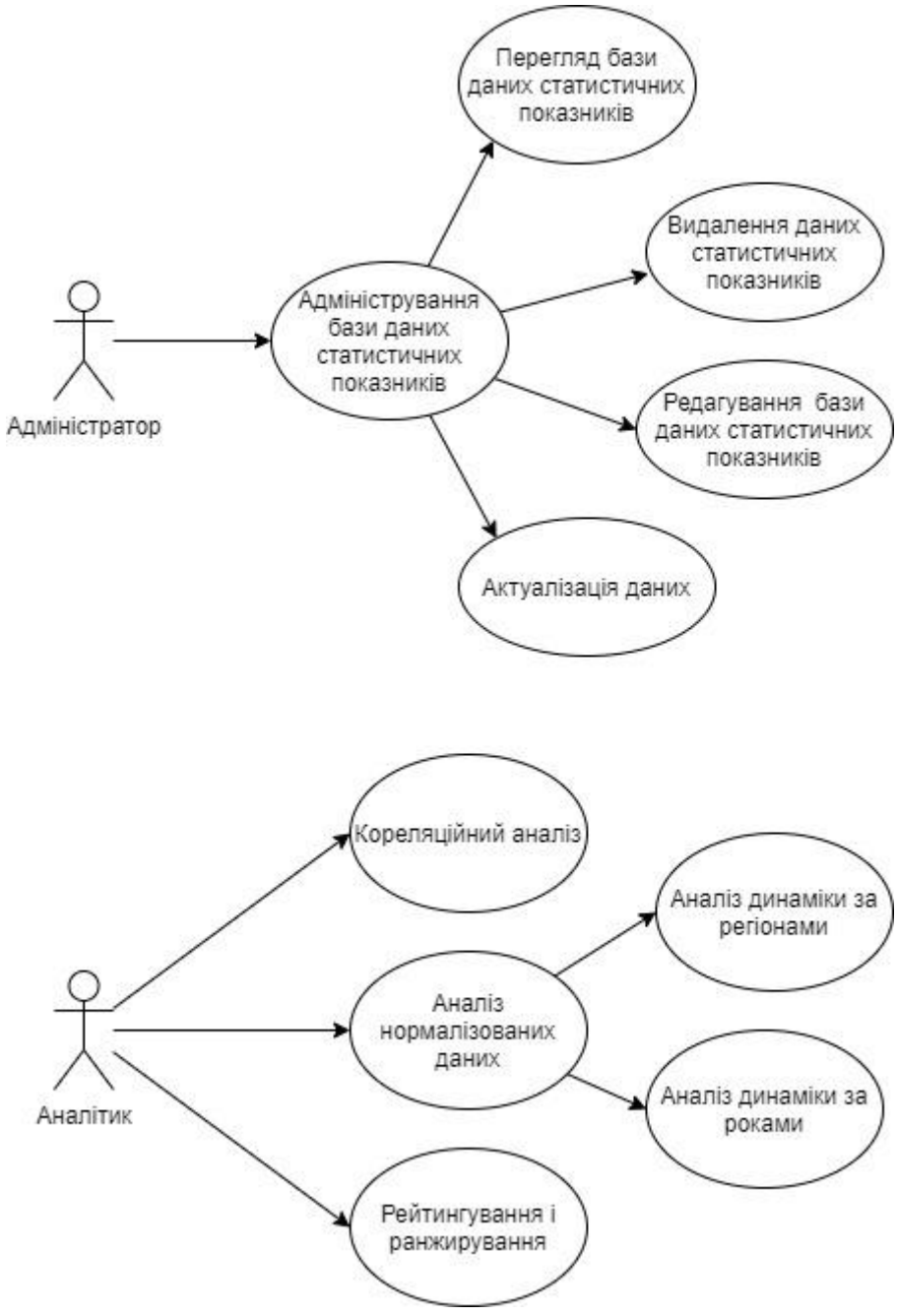


Рисунок 4.1 – Діаграма прецедентів (опис ролей користувачів)

4.2 Концептуальна модель БД

Реляційна база даних містить чотири таблиці, які створюють умови для зберігання, та отримання швидкого доступу до даних. Концептуальна модель зображена на рисунку 4.2.

Таблиця “Регіон” зберігає дані про кожен регіон.

Таблиця “Рік” зберігає дані про усі роки, для яких надійшла актуальна інформація.

Таблиця “Інформація” містить дані індикаторів, повна назва, скорочення, яке використовується в програмі та одиниці вимірювання.

Таблиця “Набір даних індикаторів” містить у собі значення усіх індикаторів для окремого регіону за обраний рік.

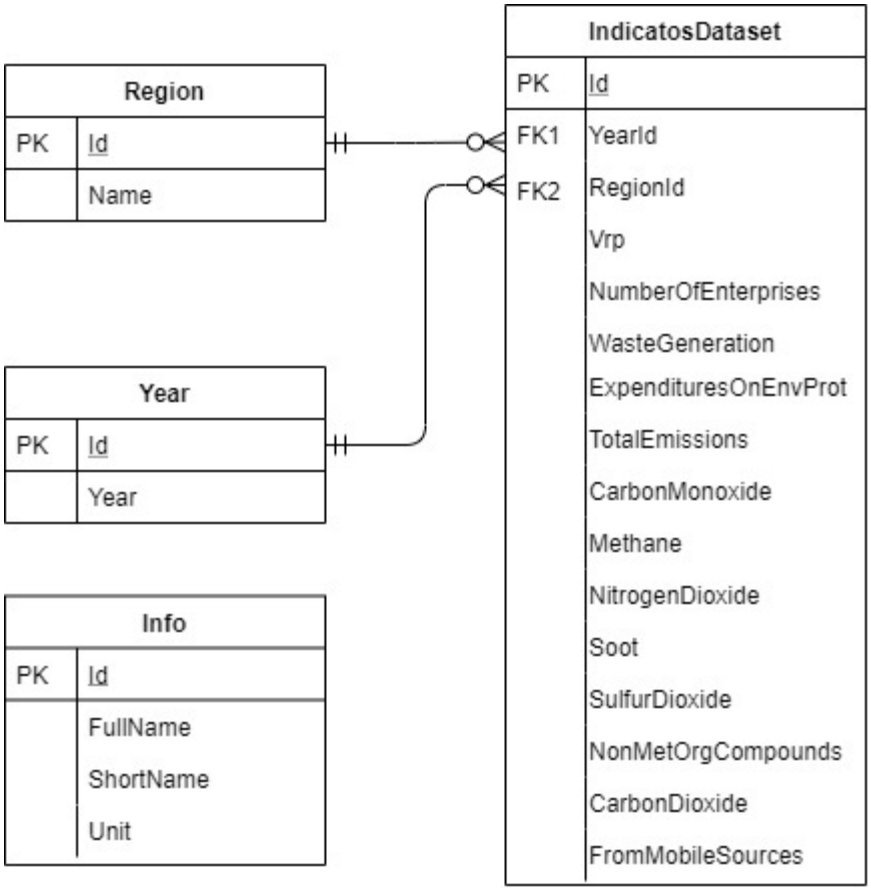


Рисунок 4.2 – Концептуальна модель БД

4.3 Опис таблиць БД

Для доступу до бази даних використовується ORM Entity Framework Core. Для кожної таблиці створюється клас сутності на мові C# з публічними властивостями, який буде відображати таблицю БД у вигляді об’єкту.

Клас контексту даних забезпечує інтерфейс і CRUD операцій над даними, що зберігаються в базі даних. Клас DbSet відповідає таблиці в базі даних, поле моделі являє собою значення стовбця таблиці.

Детальна інформація про їх структури (ім’я, тип і розмір, опис) приведена у таблицях 4.1 – 4.4.

Таблиця 4.1. Структура таблиці “Набір даних індикаторів”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	int	Первинний ключ
YearId	int	Зовнішній ключ
RegionId	int	Зовнішній ключ
VRP	real	ВРП
VRPperPerson	real	ВРП на особу
NumOfLargeInterPrises	real	Кількість великих підприємств
IndustryIndex	real	Індекс промисловості
TotalEmission	real	Обсяг забруднюючих речовин
CarbonMonoxide	real	Оксиду вуглецю
Methane	real	Метану
NitrogenDioxide	real	Діоксиду азоту
NitricOxide	real	Оксиду азоту
Soot	real	Сажі
SulfurDioxide	real	Діоксиду сірки
NonMethaneCompounds	real	Неметанових легких органічних сполук
CarbonDioxide	real	Діоксиду вуглецю

Таблиця 4.2. Структура таблиці “Регіон”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	PK, int	Первинний ключ
Name	nvarchar(100)	Назва регіону

Таблиця 4.3. Структура таблиці “Рік”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	PK, int	Первинний ключ
Year	int	Номер року

Таблиця 4.4. Структура таблиці “Інформація”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	PK, int	Первинний ключ
FullName	nvarchar(200)	Повна назва
ShortName	nvarchar(50)	Скорочення
Unit	nvarchar(100)	Одиниці виміру

4.4 Опис основних модулів

Програма складається з наступних основних модулів:

- модуль авторизації за ролями;
- модуль адміністрування БД;
- модуль регіональної статистики;
- модуль річної статистики;
- модуль кореляційного аналізу.

Модуль авторизації за ролями дозволяє розмежувати доступ до ресурсів в залежності від групи, до якої належить користувач.

Для впровадження такої авторизації необхідно підключити окрему базу даних, яка буде зберігати дані користувачів. Така база даних буде мати дві таблиці “Користувач” і “Роль”. В нашому випадку ролі всього дві:

1) користувач;

2) адміністратор.

Для реалізації необхідно буде встановити аутентифікаційні кукі. При вході в програму користувачу буде надано доступ у ролі “Користувача”. Окремо для кожного методу контролера треба буде вказати обмеження за ролями.

Модуль адміністрування БД має обмеження доступу (тільки для адміністратора) і дає наступні можливості:

- переглядати дані усіх таблиць, з можливістю сортування за будь-яким стовпцем таблиці(як за зростанням так і за спаданням), фільтрації та посторінкової навігації;
- редагувати записи таблиці;
- видаляти записи з таблиці.

Доступ до **модуль регіональної статистики** є як, у користувача так і у адміністратора. Модуль надає наступні можливості:

- можливість обрати регіон, для якого буде виконано аналіз;
- переглянути табличні дані відповідного регіону;
- переглянути динаміку викидів основних забруднюючих речовин (з зазначенням середнього значення) у вигляді діаграми комбінованої з графіком
- перегляд динаміки інших викидів у графічному вигляді;
- перегляд динаміки зміни індикаторів антропогенного впливу у графічному вигляді.

Модуль річної статистики також доступний, як для користувача так і для адміністратора. Модуль надає наступні можливості:

- можливість обрати рік, для якого буде виконано аналіз;
- переглянути табличні дані відповідного року;
- переглянути динаміку загальної кількості викидів основних забруднюючих речовину вигляді діаграми по кожному регіону;

- перегляд відсоткового складу викидів основних забруднюючих речовин за регіонами;
- перегляд динаміки зміни інших викидів у вигляді діаграм;
- перегляд динаміки зміни індикаторів антропогенного впливу у вигляді діаграм.

Модуль кореляційного аналізу має враховувати дані усіх індикаторів за усі роки, при актуалізації БД, матриця кореляції повинна перераховуватись. Доступ до даного модуля є як, у користувача так і у адміністратора. Модуль надає наступні можливості візуально, завдяки кольорам, оцінити силу зв’язку між показниками. Також при наведенні на кольорове відображення можна було побачити назви відповідних індикаторів та значення кореляції між ними.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

В цьому розділі приведені системні вимоги для роботи з додатком та сценарії роботи користувача з ним.

5.1 Системні вимоги

Програмний застосунок працює з використанням веб-технологій, він потребує від користувача підключення до інтернету та встановлений браузер. Для використання необхідний веб-браузер, який підтримує актуальні веб-стандарти.

Для забезпечення коректної та безвідмовної роботи інформаційної системи моніторингу персональний комп’ютер повинен мати процесор не гірше, ніж Intel ® Pentium ® / Celeron ® / Xeon™ або з тактовою частотою не менше 1,8 GHz або AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron ™ для користувачів процесорів від фірми AMD. Також комп’ютеру користувача повинно бути доступно не менше 2 Gb оперативної пам’яті та графічне ядро не гріше, ніж Intel ® HD Graphics 2000, що еквівалентно графічним картам з об’ємом пам’яті не менше, ніж 128 Mb.

5.2 Робота користувача з програмним продуктом

Для того щоб увійти в систему користувачу необхідно пройти авторизацію. Для цього в діалоговому вікні, зображеному на рисунку 5.1, необхідно ввести свій логін і пароль – для адміністраторів, а для аналітиків – натиснути кнопку «Увійти як користувач» (Програма введе дані для входу для користувача). Далі слід натиснути кнопку «Вхід». Якщо користувач ввів вірні дані – система його успішно авторизує та надасть доступ до функцій програми.

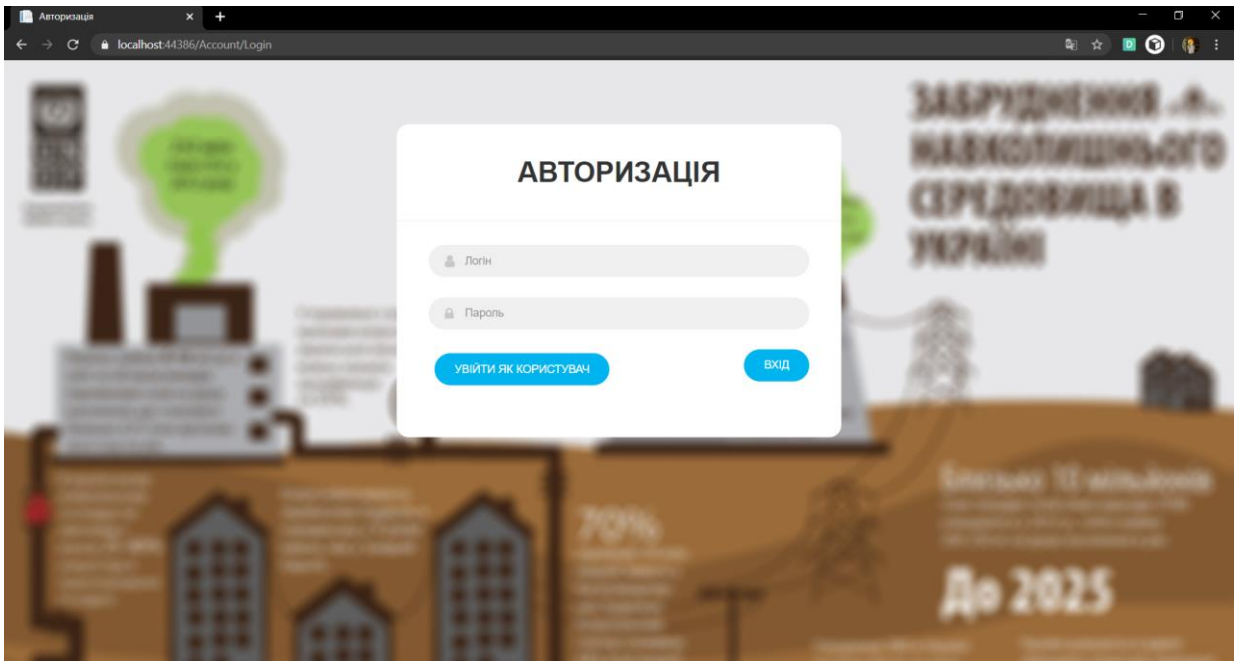


Рисунок 5.1 – Сторінка авторизації

Після успішної авторизації, ми потрапляємо на головну сторінку, де є можливість ознайомитись з основними принципами стратегії НВР, отримати посилання на першоджерела стратегії, переглянути список показників, які використовує програма для моніторингу.

Головне меню буде доступно з будь-якого місця програми та буде розташоване у верхній частині сторінки (рисунок 5.2).

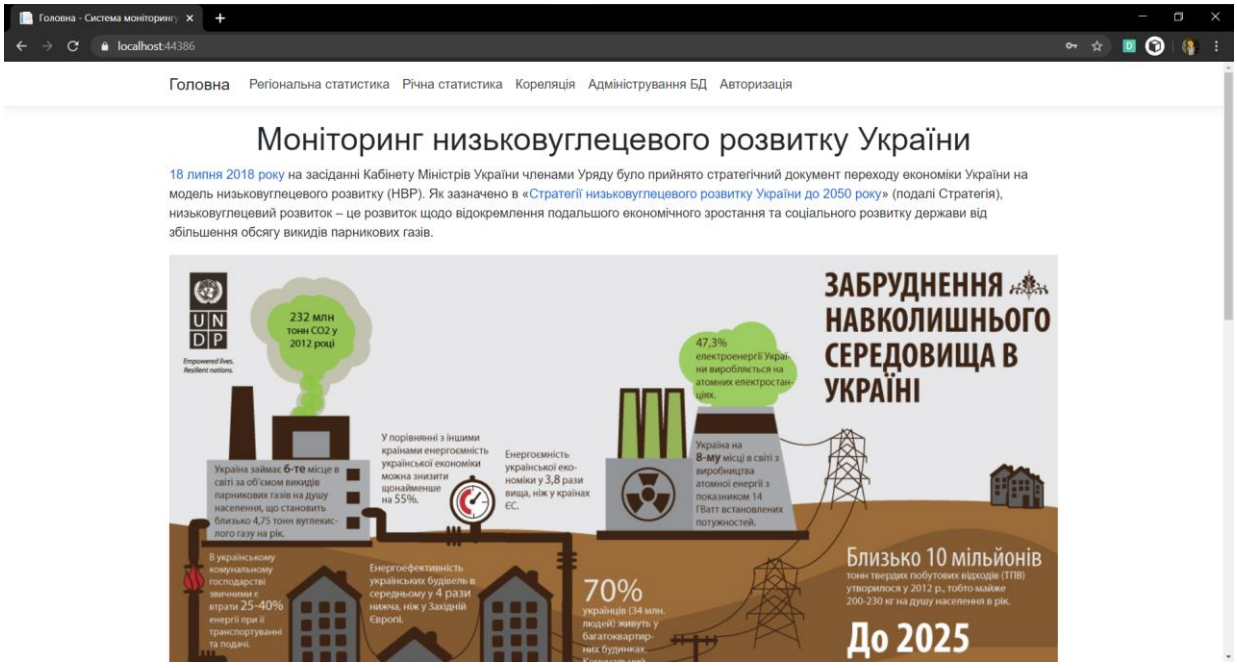


Рисунок 5.2 – Головна сторінка

Також, елементом, який буде незмінним у будь-якій частині програми буде колонтитул, на якому вказано посилання на ліцензію використання даних, та зазначено посилання на джерела даних (рисунок 5.3).

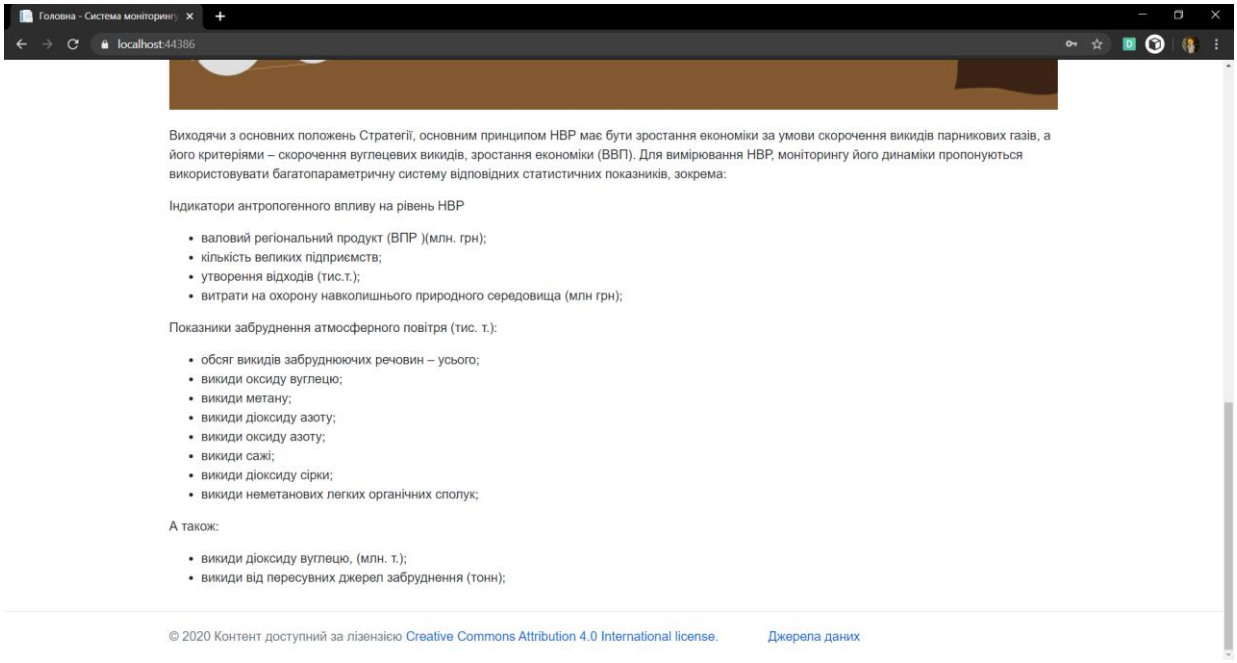


Рисунок 5.3 – Колонтитул

Якщо ми перейдемо за посиланням “Джерела даних”, ми побачимо сторінку, на якій зазначені джерела усіх даних, як того вимагає постачальних даних, а саме “Державна служба статистики України” (рисунок 5.4).

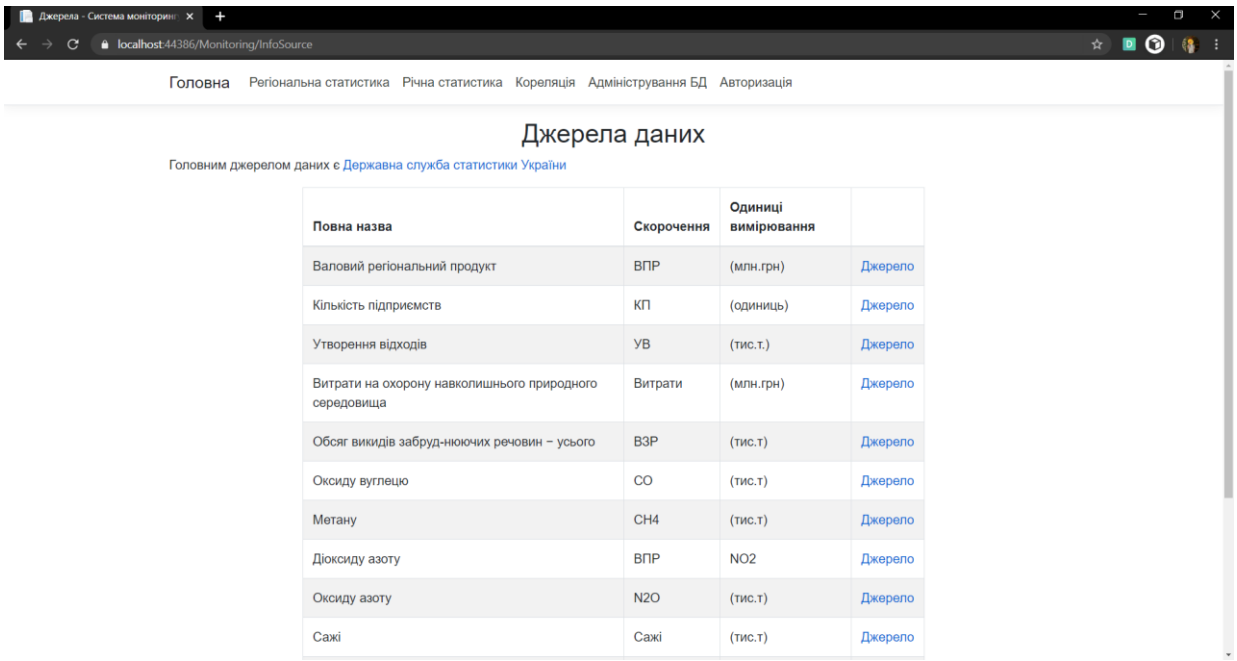
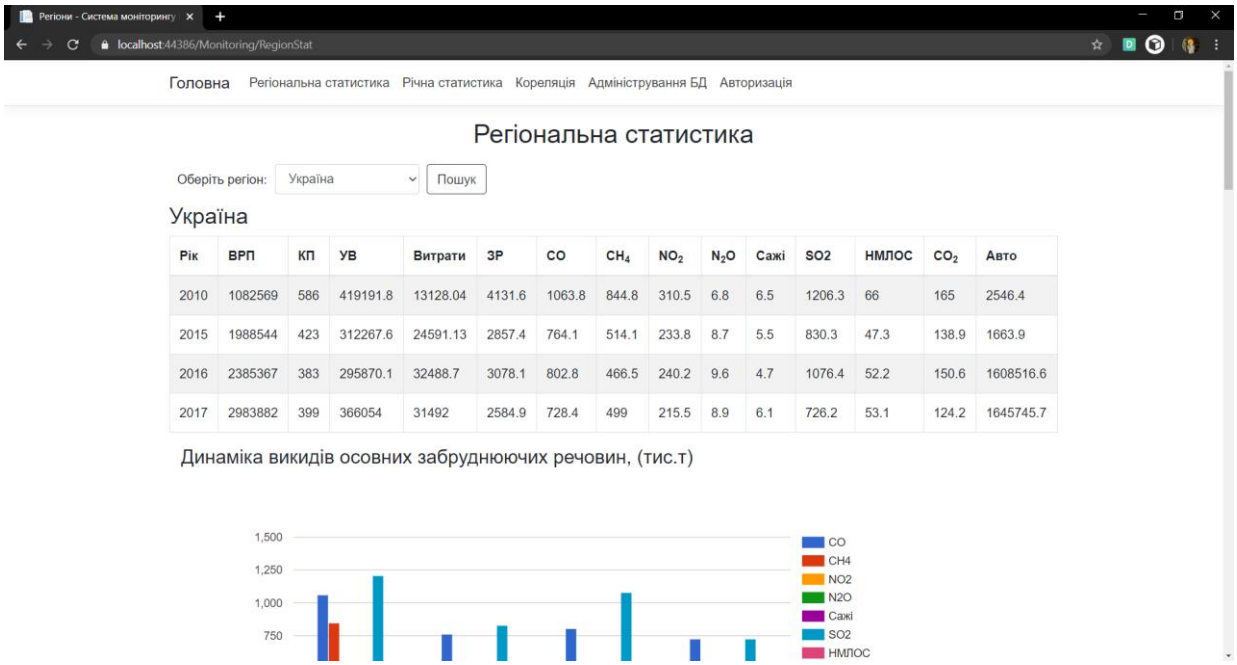


Рисунок 5.4 – Джерела даних

Для зручності, кожен показник має окреме посилання. Використовуючи головне меню, ми можемо перейти до пункту “Регіональна статистика”. Де, за замовчуванням, обрані дані з усієї України (рисунок 5.5).



Рік	ВРП	КП	УВ	Витрати	ЗР	CO	CH_4	NO_2	N_2O	Сажі	SO2	НМЛОС	CO_2	Авто
2010	1082569	586	419191.8	13128.04	4131.6	1063.8	844.8	310.5	6.8	6.5	1206.3	66	165	2546.4
2015	1988544	423	312267.6	24591.13	2857.4	764.1	514.1	233.8	8.7	5.5	830.3	47.3	138.9	1663.9
2016	2385367	383	295870.1	32488.7	3078.1	802.8	466.5	240.2	9.6	4.7	1076.4	52.2	150.6	1608516.6
2017	2983882	399	366054	31492	2584.9	728.4	499	215.5	8.9	6.1	726.2	53.1	124.2	1645745.7

Рисунок 5.5 – Регіональна статистика

Користувач може обрати будь-який регіон використовуючи список регіонів, натиснути кнопку “Пошук” та отримати дані відповідного регіону. Також, для зручності було реалізовано можливість побачити додаткову інформацію, при наведенні на його скорочену назву у таблиці (рисунок 5.6).

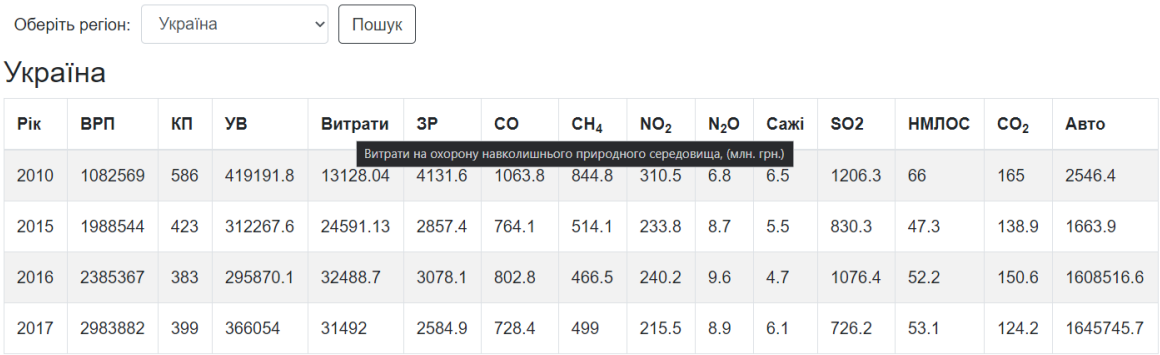


Рисунок 5.6 – Додаткова інформація

На сторінці регіональної статистики, окрім табличного представлення даних, можна переглянути динаміку основних забруднюючих речовин у вигляді діаграми комбінованої з графіком (рисунок 5.7).

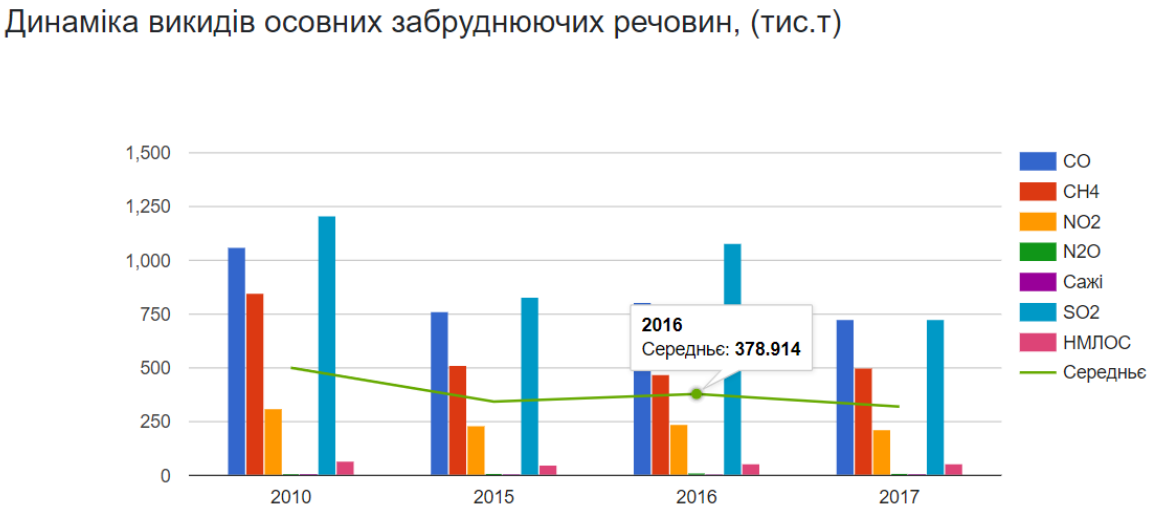


Рисунок 5.7 – Комбінована діаграма динаміки викидів

Гортаючи сторінку вниз можна переглянути динаміку інших викидів, та динаміку зміни індикаторів антропогенного впливу (рисунок 5.8).

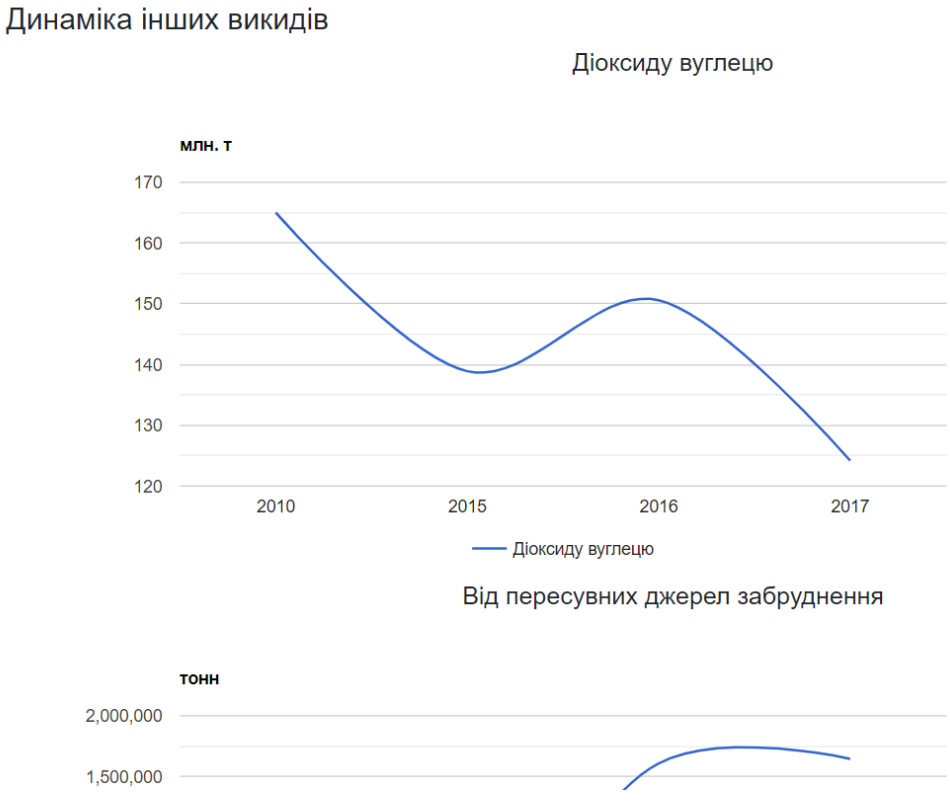
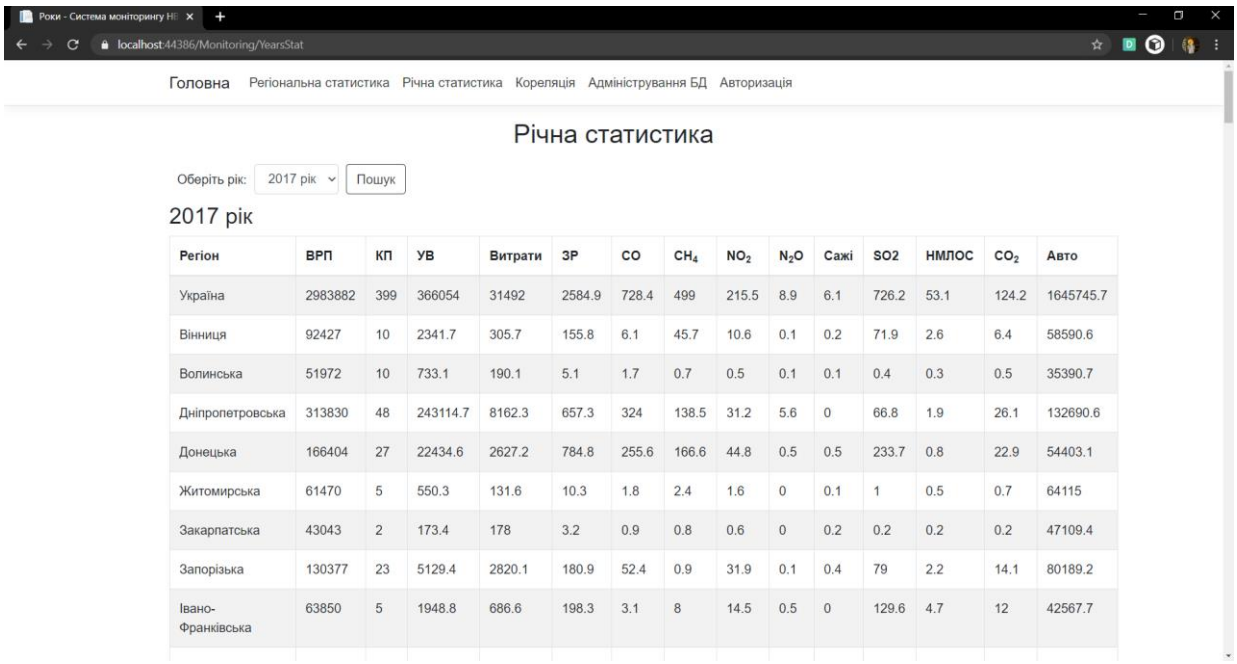


Рис 5.8 – Графік динаміки інших викидів

Для перегляду річної статистики, необхідно обрати пункт “Річна статистика” в головному меню (рисунок 5.9).



Регіон	ВРП	КП	УВ	Витрати	ЗР	CO	CH_4	NO_2	N_2O	Сажі	SO2	НМЛОС	CO_2	Авто
Україна	2983882	399	366054	31492	2584.9	728.4	499	215.5	8.9	6.1	726.2	53.1	124.2	1645745.7
Вінниця	92427	10	2341.7	305.7	155.8	6.1	45.7	10.6	0.1	0.2	71.9	2.6	6.4	58590.6
Волинська	51972	10	733.1	190.1	5.1	1.7	0.7	0.5	0.1	0.1	0.4	0.3	0.5	35390.7
Дніпропетровська	313830	48	243114.7	8162.3	657.3	324	138.5	31.2	5.6	0	66.8	1.9	26.1	132690.6
Донецька	166404	27	22434.6	2627.2	784.8	255.6	166.6	44.8	0.5	0.5	233.7	0.8	22.9	54403.1
Житомирська	61470	5	550.3	131.6	10.3	1.8	2.4	1.6	0	0.1	1	0.5	0.7	64115
Закарпатська	43043	2	173.4	178	3.2	0.9	0.8	0.6	0	0.2	0.2	0.2	0.2	47109.4
Запорізька	130377	23	5129.4	2820.1	180.9	52.4	0.9	31.9	0.1	0.4	79	2.2	14.1	80189.2
Івано-Франківська	63850	5	1948.8	686.6	198.3	3.1	8	14.5	0.5	0	129.6	4.7	12	42567.7

Рисунок 5.9 – Річна статистика

Нище на сторінці річної статистики, можна побачити діаграму загальної кількості викидів за обраний рік за регіонами (рисунок 5.10).

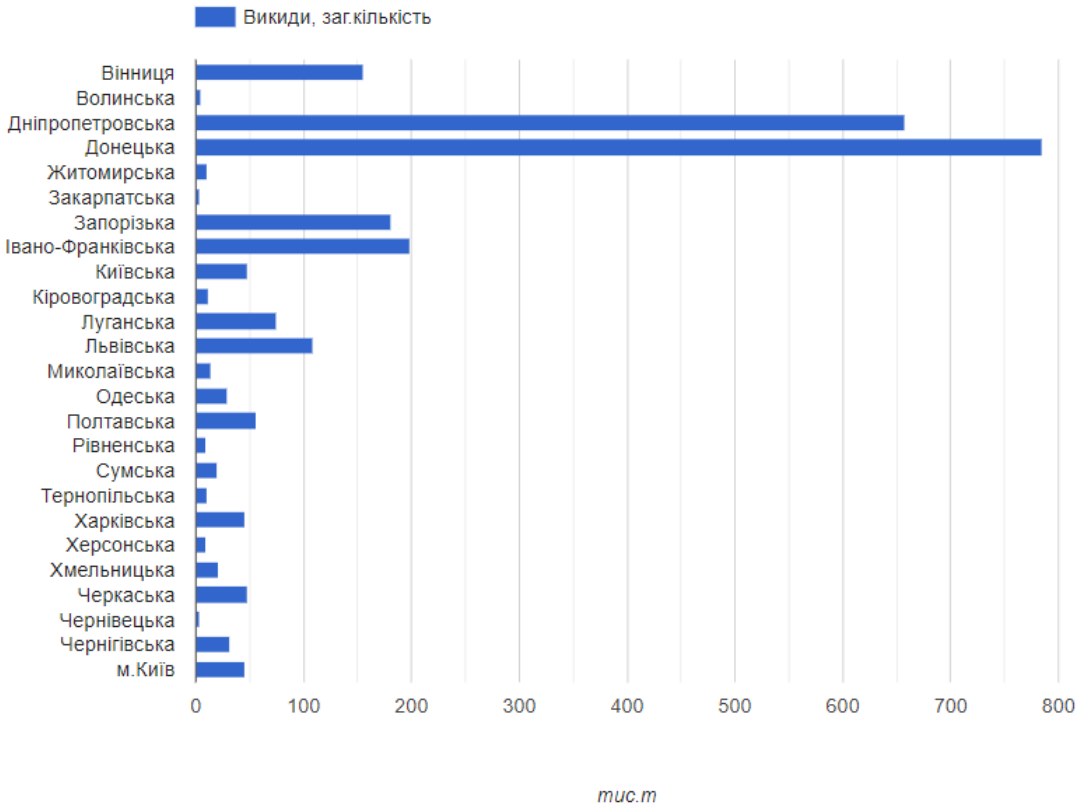


Рисунок 5.10 – Регіональна статистика – загальна кількість викидів

Ще нижче зображені викиди основних забруднюючих речовин у відсотках % (рисунок 5.11).

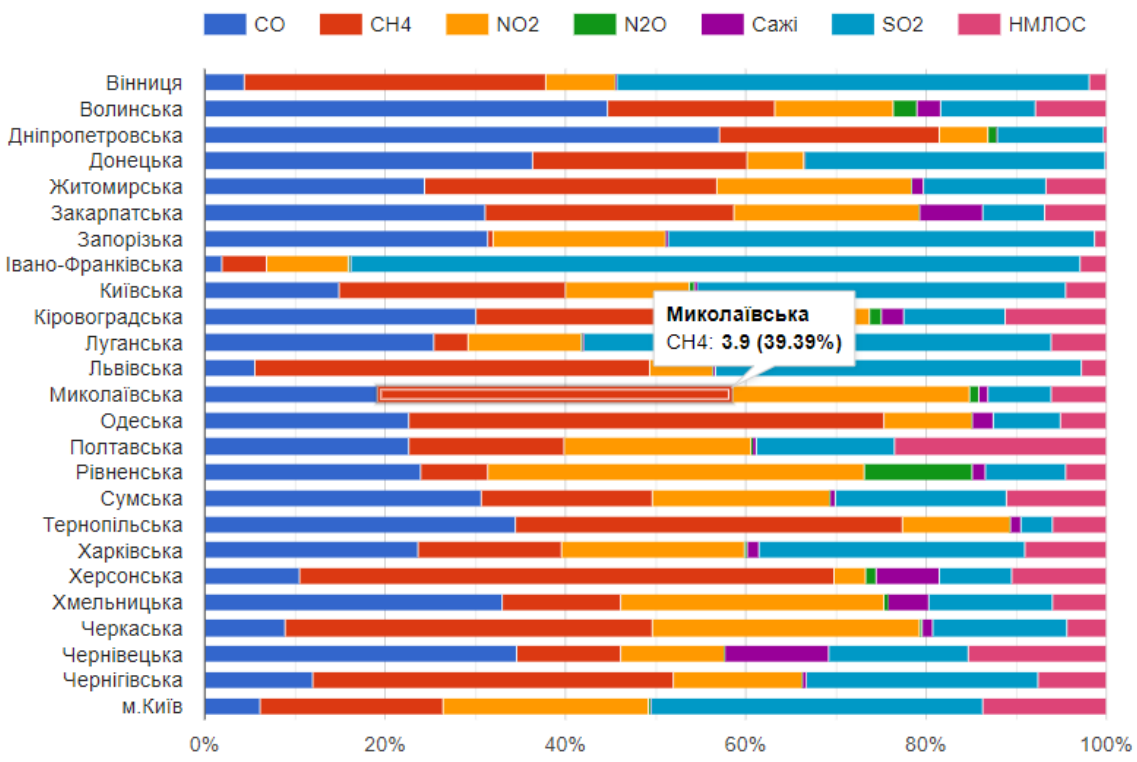


Рисунок 5.11 – Річна статистика – викиди у відсотках

При натисканні на відповідний проміжок можна побачити назву області, назву, кількість та відсоток від загальної кількості викидів.

Ще нижче зображені діаграми інших викидів та індикаторів антропогенного впливу (рисунок 5.12).

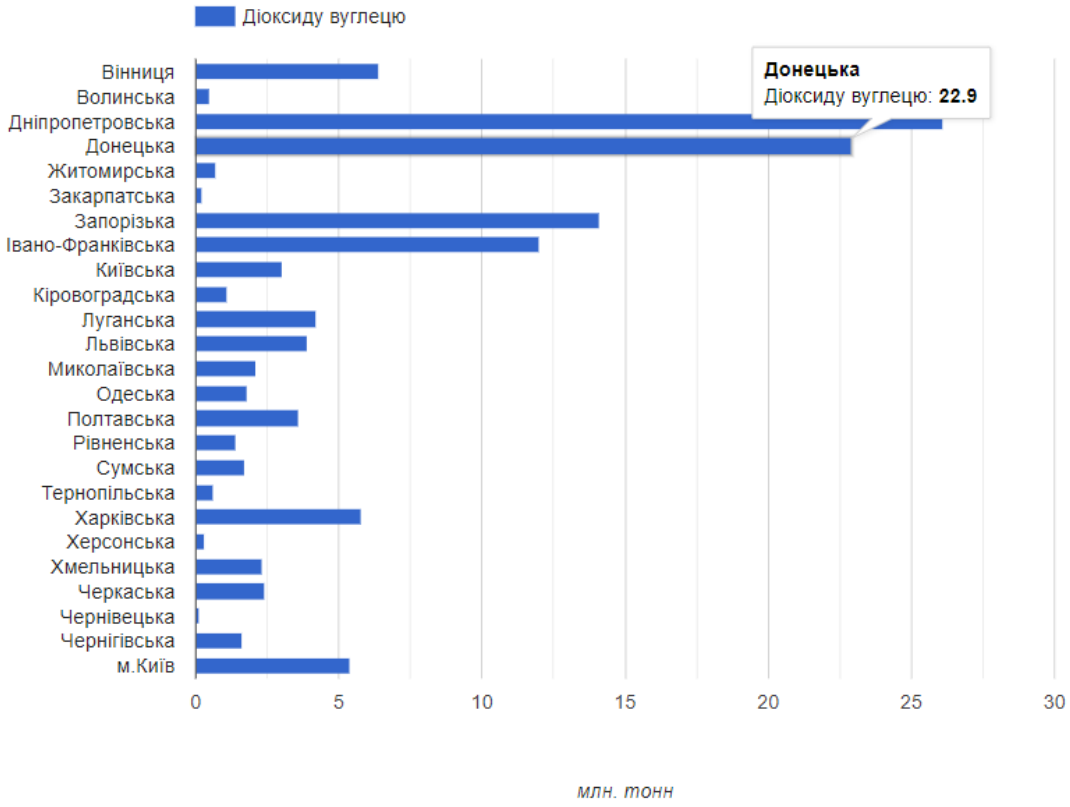


Рисунок 5.12 – Річна статистика – діаграма викидів діоксиду вуглецю

Для переходу на кореляційний аналіз обираємо пункт головного меню – “Кореляція”. На сторінці, використовуючи кольорову шкалу сили зв’язку, наводячи на елемент матриці, можна побачити, між якими індикаторами та яке значення коефіцієнту кореляції (рисунок 5.13).

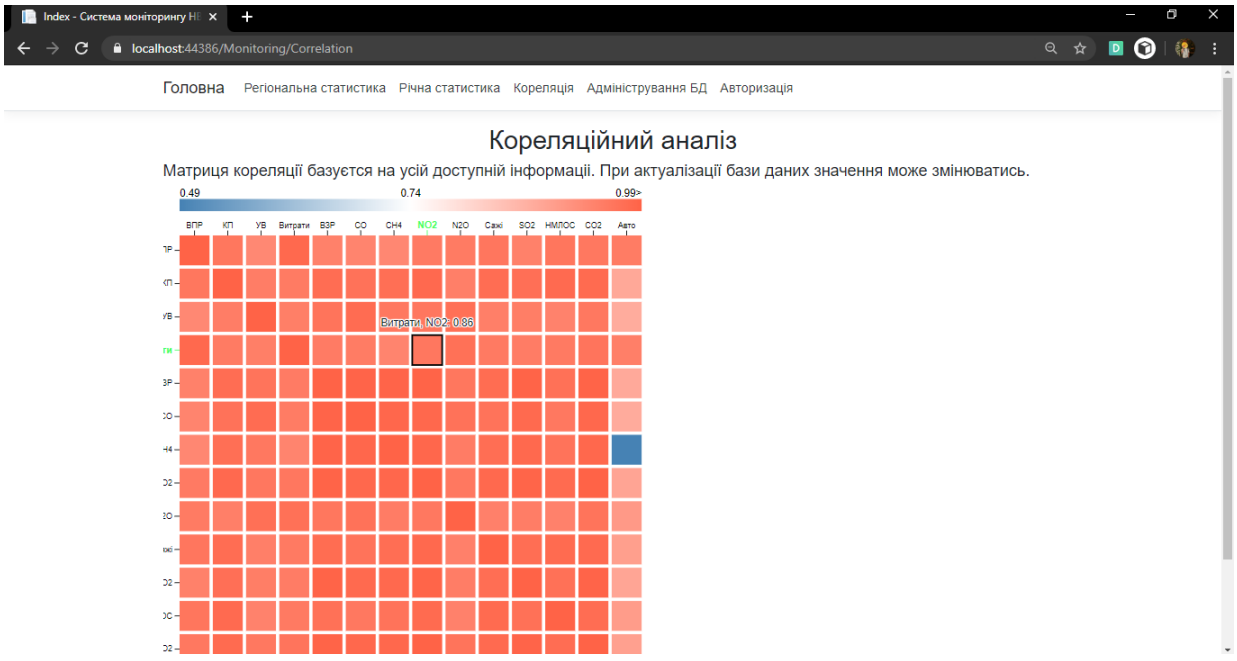


Рисунок 5.13 – Кореляційний аналіз

Для переходу у меню адміністрування БД, необхідно обрати відповідний пункт у головному меню. У випадку, якщо у вас не має відповідних прав адміністратора – вас буде повернуто у меню авторизації. Авторизувавшись, як адміністратор, ви можете перейти до меню адміністрування (рисунок 5.14).

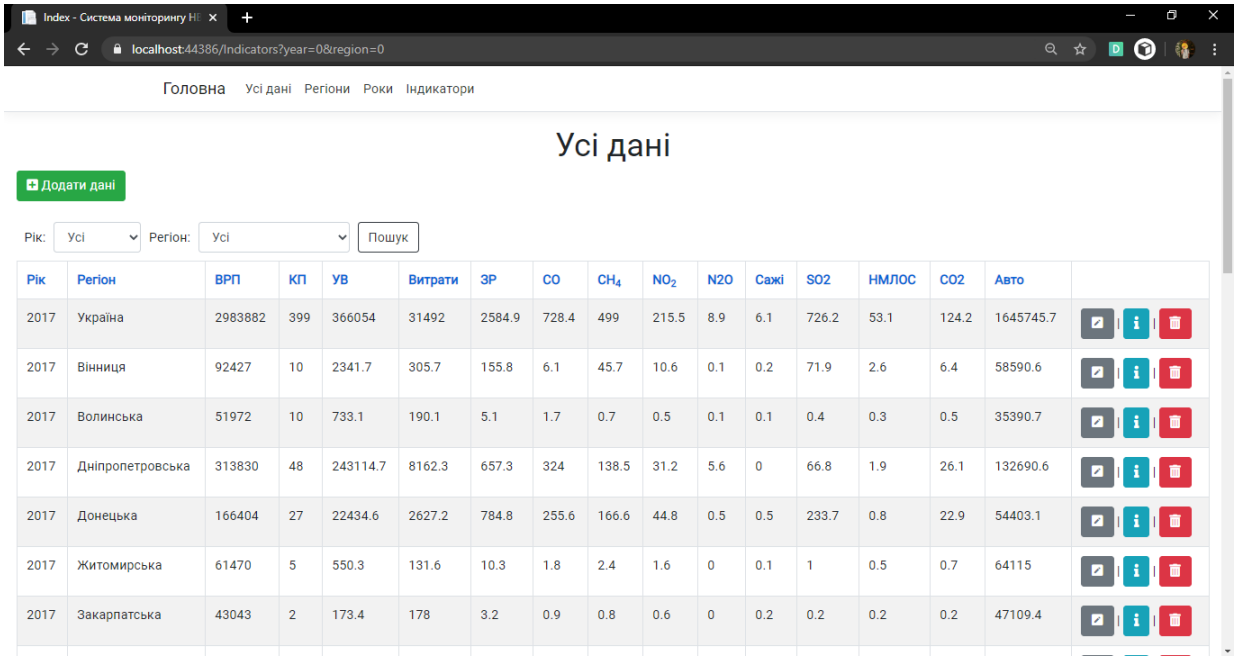


Рисунок 5.14 – Адміністрування БД

Першу, що можна на сторінці побачити – це повна таблиця регіональних за всіма роками. Для зручності можемо користуватись фільтрацією, обравши необхідний рік, регіон, або залишити значення “Усі”. Натиснувши кнопку пошук ми отримаємо дані за заданими параметрами фільтру (рисунок 5.15).

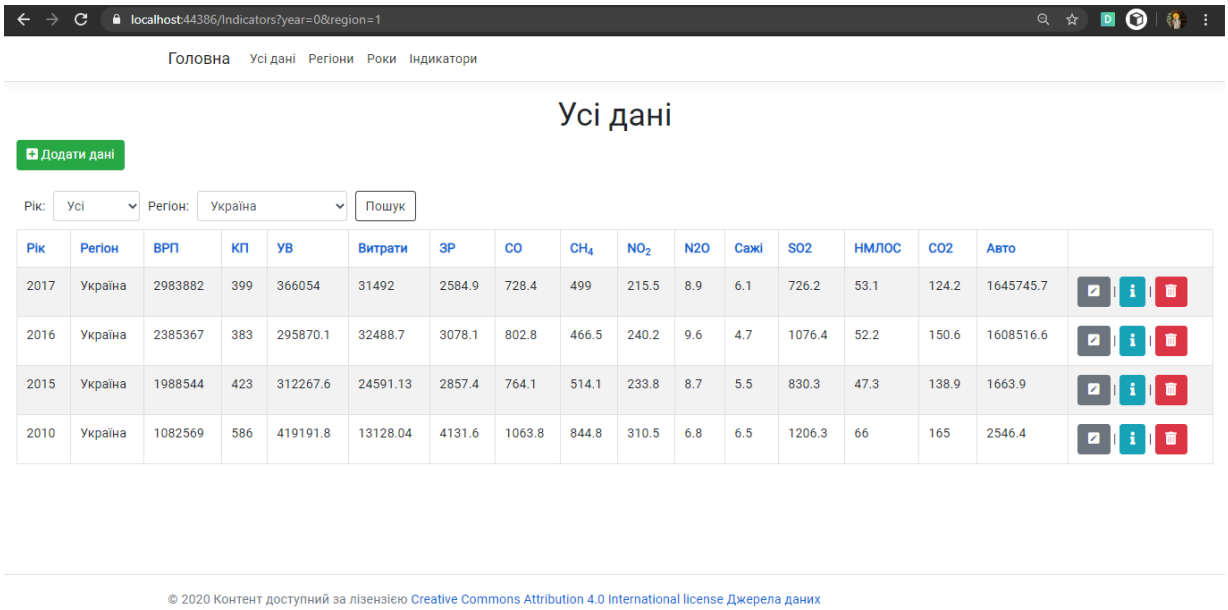


Рисунок 5.15 – Фільтрація

Також є функція сортування за будь-яким стовбцем таблиці, для цього потрібно натиснути на заголовок стовбця та таблиця буде відсортована за спаданням значення цього стовпця, якщо натиснути повторно на цей самий стовпець – таблиця буде відсортована за спаданням значення цього стовпця.

Щоб додати дані до таблиці, треба натиснути на відповідну кнопку у верхньому правому куті сторінки і ми перейдемо до форми додавання нового набору даних (рисунок 5.16).

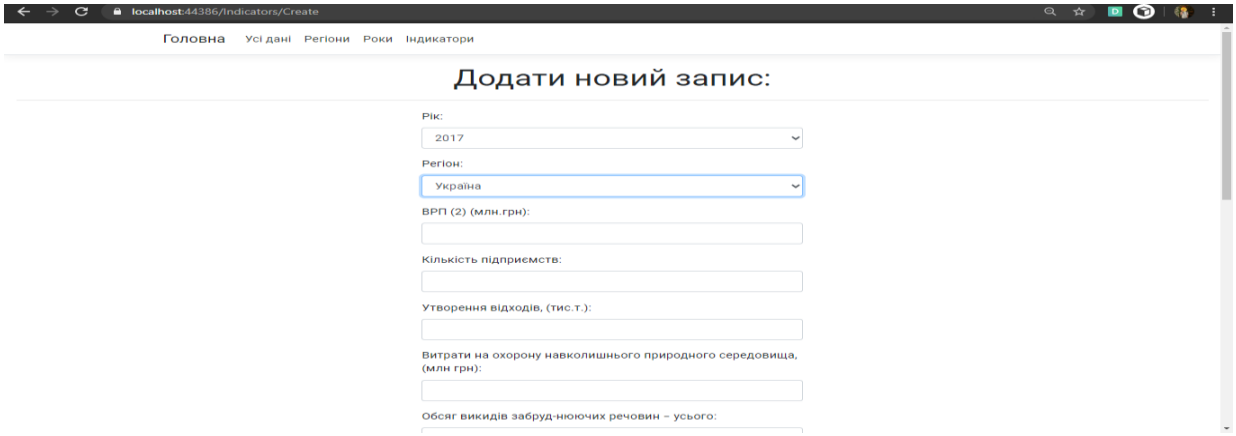


Рисунок 5.16 – Додати запис

Для того, щоб додати новий запис, треба обрати необхідний рік, регіон, ввести значення усіх показників та натиснути кнопку “Додати” внизу сторінки. Якщо необхідно додати дані за рік, або регіон яких ще немає у базі даних, необхідно спочатку перейти до таблиці “Регіони” або “Роки” та додати нові дані (рисунок 5.17).

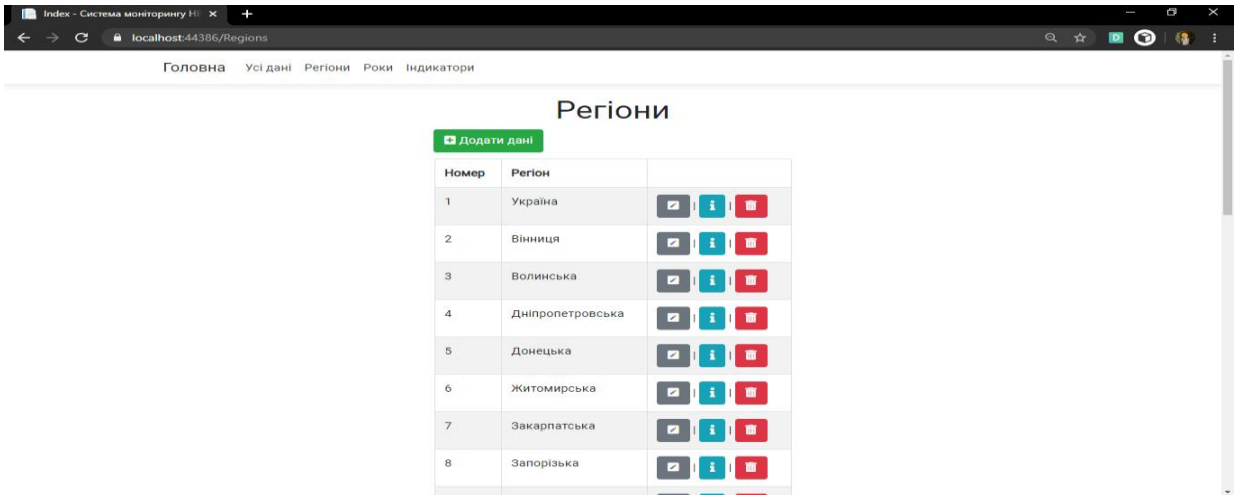


Рисунок 5.17 – Таблиця “Регіони”

Для редагування, перегляду детальної інформації або видалення даних – необхідно обрати відповідну кнопку напроти строки (рисунок 5.18).

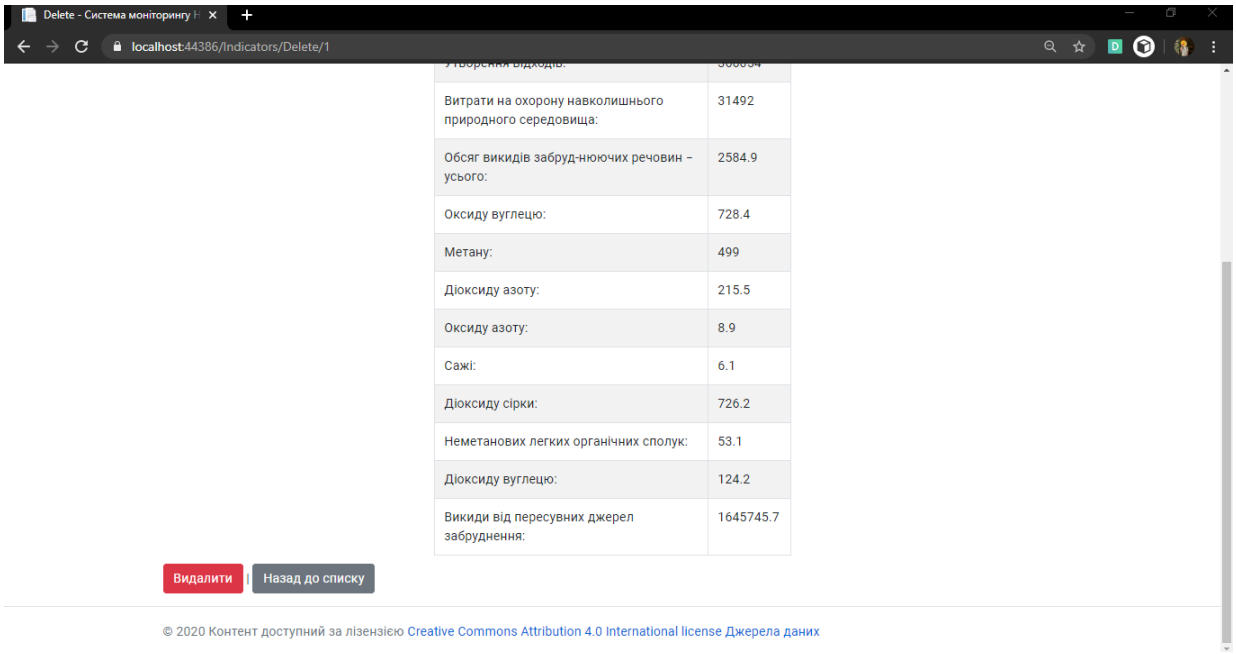


Рисунок 5.18 – Видалення запису

Будь-які зміни таблиць в меню “Адміністрування БД” одразу надходять до БД. Тобто, якщо додати нові дані, і повернутись у меню статистики або кореляційного аналізу, одразу можна побачити нові статистичні дані та перераховану матрицю кореляції.

ВИСНОВКИ

В ході виконання даної роботи було розроблено багатопараметричну систему моніторингу низьковуглецевого розвитку України.

Програму було написано на мові програмування C# з використанням графічного веб-інтерфейсу.

Програму можна використовувати для вимірювання рівень НВР, на основі статистичних даних, та займатись моніторингом його динаміки.

В ході роботи було отримано, що були використані для створення даного програмного забезпечення (середовища розробки Microsoft Visual Studio 2019, ASP.NET Core Framework та засобів створення веб-інтерфейсів: HTML5, CSS, JavaScript).

Для роботи з даним програмним забезпеченням необхідний лише комп’ютер середньої потужності.

Користувачу буде надана можливість опрацьовувати дані різних показників та індикаторів, усіх регіонів за останні роки, прогнозувати варіанти подальшого низьковуглецевого розвитку і аналізувати ефективність стратегії відокремлення подальшого економічного зростання та соціального розвитку від збільшення обсягу викидів парникових газів. Користувач може зберігати звіти на особистий комп’ютер.

Програма дає можливість доповнювати базу даних статистичними показниками за новий період.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стратегія низьковуглецевого розвитку України до 2050 року – URL: https://menr.gov.ua/news/31815.html.

2. Урядовий портал. Єдиний веб-портал органів виконавчої влади України. «Остап Семерак: Уряд України підтримав Стратегію низьковуглецевого розвитку України до 2050 року» – URL: https://www.kmu.gov.ua/news/ostap-semerak-uryad-ukrayini-pidtrimav-strategiyu-nizkovuglecevogo-rozvitku-ukrayini-do-2050-roku.

3. Національна доповідь «Цілі Сталого Розвитку: Україна» URL: http://un.org.ua/images/SDGs_NationalReportUA_Web_1.pdf.

4. Караєва Н. В., Варава І. А. Еколого-економічна оптимізація виробництва: методи та засоби статистичного прогнозування [Електронний ресурс] : конспект лекцій. – Київ : НТУУ «КПІ», 2016. 80 с. – URL: http://ela.kpi.ua/bitstream/123456789/15377/1/Stat_prognoz.pdf.

5. Human Development Reports UNDP "Human Development Indices and Indicators: 2018 Statistical Update". URL: http://hdr.undp.org/en/2018-update.

6. Статистичний збірник "Регіональний людський розвиток" за 2017 рік / відповідальний за випуск О.О. Кармазіна. URL: http://www.ukrstat.gov.ua.

7. Сайт о программировании. C# / .NET / Руководство по ASP.NET Core 3 [Електронний ресурс] / Е.Попов - URL: https://metanit.com/sharp/aspnet5/

8. Уэйт М. Язык C#. Руководство для начинающих. / М. Уэйт, С. Прага, Д. Мартин. – М. : Мир, 1995. – 521с.

9. Троэлсен Э. C# и платформа .NET. Библиотека программиста / Э. Троэлсен. – СПб. : Питер, 2004. – 126 с.

10. Сайт о программировании. C# / .NET / Руководство по MS SQL Server 2017 [Електронний ресурс] / Е.Попов – URL: https://metanit.com/sharp/aspnet5/.

11. Адам Фримен — ASP.NET Core MVC с примерами на C# для профессионалов [Електронний ресурс]. – 2017. – URL: https://bit.ly/2JbSgHe.

12. Сайт о программировании. C# / .NET / Entity Framework Core [Електронний ресурс] / Е.Попов – URL: https://metanit.com/sharp/entityframeworkcore/.

ДОДАТОК 1

Система моніторингу рівня низьковуглецевого розвитку України

Специфікація

УКР.НТУУ «КПІ ім. І.Сікорського».ТМ-61

Аркушів 4

2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ім. І.Сікорського».ТМ4110_18Б 81-1	Записка_Анненков_ТМ-61.docx	Пояснювальна записка
Компоненти		
	Year.cs	Клас (entity), який представляє рік
	Region.cs	Клас (entity), який представляє регіон
	IndicatorInfo.cs	Клас (entity), який представляє інформацію про індикатори
	IndicatorsModel.cs	Клас (entity), який представляє індикатори
	IndicatorsDbContext.cs	Клас, який забезпечує взаємодію з базою даних
	SortViewModel.cs	Модель сортування
	SortState.cs	Перелік(enum), який використовується для сортування
	PageViewModel.cs	Модель пагінації(посторінкової навігації)
	IndexViewModel.cs	Загальна модель, яка об’єднує фільтрацію, сортування та пагінацію
	FilterViewModel.cs	Модель фільтрації
	ErrorViewModel.cs	Модель виводу помилок
	ChartFilterViewModel.cs	Модель сортування даних діаграми
	User.cs	Клас (entity), який представляє користувача
	Role.cs	Клас (entity), який представляє рол користувача
	LoginModel.cs	Модель авторизації

	AutorisationContext.cs	Клас-контролер, який забезпечує взаємодію з базою даних користувачів
	AccountController.cs	Клас-контролер, який реалізує реєстрацію, вхід, вихід користувачів
	HomeController.cs	Клас-контролер, який реалізує перехід на головну сторінку
	IndicatorInfoesController.cs	Клас-контролер, який забезпечує взаємодію з таблицею “Інфо”
	IndicatorsController.cs	Клас-контролер, який забезпечує взаємодію з таблицею “Індикатори”
	MonitoringController.cs	Клас-контролер, який виконує основні операції з даними для відображення
	RegionsController.cs	Клас-контролер, який забезпечує взаємодію з таблицею “Регіони”
	YearsController.cs	Клас-контролер, який забезпечує взаємодію з таблицею “Роки”
	_Layout.cshtml	Представлення, яке реалізує header та footer програми
	_LayoutForDB.cshtml	Представлення, яке реалізує header та footer програми для адміністрування БД
	Home/Index.cshtml	Представлення, яке відображає головну сторінку
	Login.cshtml	Представлення, яке відображає форму входу

	Index.cshtml	Представлення, яке відображає усі дані таблиці (наявне для кожної таблиці)
	Create.cshtml	Представлення, для запису нових даних до таблиці(наявне для кожної таблиці)
	Delete.cshtml	Представлення, для видалення даних до таблиці(наявне для кожної таблиці)
	Details.cshtml	Представлення, для відображення деталей даних таблиці(наявне для кожної таблиці)
	Edit.cshtml	Представлення, для редагування даних таблиці(наявне для кожної таблиці)
	Correlation.cshtml	Представлення, для відображення кореляційного аналізу
	InfoSource.cshtml	Представлення, для відображення інформації та джерел індикаторів
	RegionStat.cshtml	Представлення, для відображення регіональної статистики
	YearsStat.cshtml	Представлення, для відображення річної статистики

ДОДАТОК 2

Система моніторингу рівня низьковуглецевого розвитку України

Текст програмного коду

УКР.НТУУ «КПІ ім. І.Сікорського».ТМ4110_18Б 12-23

Аркушів 13

2020

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using LCMonitoringSystem3.Models;
using Microsoft.AspNetCore.Authorization;
using System.Security.Claims;

namespace LCMonitoringSystem3.Controllers
{
    public class IndicatorsController : Controller
    {
        private readonly IndicatorsDbContext _context;

        public IndicatorsController(IndicatorsDbContext context)
        {
            _context = context;
        }

        // GET: Indicators
        [Authorize(Roles = "admin")]
        public async Task<IActionResult> Index(int? year, int? region, int
page = 1, SortState sortOrder = SortState.YearDesc)
        {
            int pageSize = 28;
```

-3-

```
//Фильтрация
        IQueryable<IndicatorsModel> indicators =
  _context.Indicators.Include(i => i.Region).Include(i => i.Year);
        if (year != null && year != 0)
        {
            indicators = indicators.Where(p => p.YearId == year);
        }
        if (region != null && region != 0)
        {
            indicators = indicators.Where(p => p.RegionId == region);
        }

        //Сортировка
        switch (sortOrder)
        {
            case SortState.YearAsc:
                indicators = indicators.OrderBy(s => s.Year.YearNumb);
                break;
            case SortState.RegionAsc:
                indicators = indicators.OrderBy(s => s.Region.Name);
                break;
            case SortState.VrpAsc:
                indicators = indicators.OrderBy(s => s.Vrp);
                break;
            case SortState.NumberOfEnterprisesAsc:
                indicators = indicators.OrderBy(s => s.NumberOfEnterprises);
                break;
            case SortState.WasteGenerationAsc:
                indicators = indicators.OrderBy(s => s.WasteGeneration);
```

```
                break;
            case SortState.ExpendituresOnEnvProtAsc:
                indicators = indicators.OrderBy(s =>
s.ExpendituresOnEnvProt);
                break;
            case SortState.TotalEmissionsAsc:
                indicators = indicators.OrderBy(s => s.TotalEmissions);
                break;
            case SortState.CarbonMonoxideAsc:
                indicators = indicators.OrderBy(s => s.CarbonMonoxide);
                break;
            case SortState.MethaneAsc:
                indicators = indicators.OrderBy(s => s.Methane);
                break;
            case SortState.NitrogenDioxideAsc:
                indicators = indicators.OrderBy(s => s.NitrogenDioxide);
                break;
            case SortState.NitricOxideAsc:
                indicators = indicators.OrderBy(s => s.NitricOxide);
                break;
            case SortState.SootAsc:
                indicators = indicators.OrderBy(s => s.Soot);
                break;
            case SortState.SulfurDioxideAsc:
                indicators = indicators.OrderBy(s => s.SulfurDioxide);
                break;
            case SortState.NonMetOrgCompoundsAsc:
                indicators = indicators.OrderBy(s =>
s.NonMetOrgCompounds);
```

```
                break;
            case SortState.CarbonDioxideAsc:
                indicators = indicators.OrderBy(s => s.CarbonDioxide);
                break;
            case SortState.FromMobileSourcesAsc:
                indicators = indicators.OrderBy(s => s.FromMobileSources);
                break;

            case SortState.RegionDesc:
                indicators = indicators.OrderByDescending(s =>
s.Region.Name);
                break;
            case SortState.VrpDesc:
                indicators = indicators.OrderByDescending(s => s.Vrp);
                break;
            case SortState.NumberOfEnterprisesDesc:
                indicators = indicators.OrderByDescending(s =>
s.NumberOfEnterprises);
                break;
            case SortState.WasteGenerationDesc:
                indicators = indicators.OrderByDescending(s =>
s.WasteGeneration);
                break;
            case SortState.ExpendituresOnEnvProtDesc:
                indicators = indicators.OrderByDescending(s =>
s.ExpendituresOnEnvProt);
                break;
            case SortState.TotalEmissionsDesc:
```

```
                indicators = indicators.OrderByDescending(s =>
s.TotalEmissions);
                break;
            case SortState.CarbonMonoxideDesc:
                indicators = indicators.OrderByDescending(s =>
s.CarbonMonoxide);
                break;
            case SortState.MethaneDesc:
                indicators = indicators.OrderByDescending(s => s.Methane);
                break;
            case SortState.NitrogenDioxideDesc:
                indicators = indicators.OrderByDescending(s =>
s.NitrogenDioxide);
                break;
            case SortState.NitricOxideDesc:
                indicators = indicators.OrderByDescending(s =>
s.NitricOxide);
                break;
            case SortState.SootDesc:
                indicators = indicators.OrderByDescending(s => s.Soot);
                break;
            case SortState.SulfurDioxideDesc:
                indicators = indicators.OrderByDescending(s =>
s.SulfurDioxide);
                break;
            case SortState.NonMetOrgCompoundsDesc:
                indicators = indicators.OrderByDescending(s =>
s.NonMetOrgCompounds);
                break;
```

```
            case SortState.CarbonDioxideDesc:
                indicators = indicators.OrderByDescending(s =>
s.CarbonDioxide);
                break;
            case SortState.FromMobileSourcesDesc:
                indicators = indicators.OrderByDescending(s =>
s.FromMobileSources);
                break;

            default:
                indicators = indicators.OrderByDescending(s =>
s.Year.YearNumb);
                break;
        }

        //Пагинация
        var count = await indicators.CountAsync();
        var items = await indicators.Skip((page - 1) *
pageSize).Take(pageSize).ToListAsync();

        // формируем модель представления
        IndexViewModel viewModel = new IndexViewModel
        {
            PageViewModel = new PageViewModel(count, page, pageSize),
            SortViewModel = new SortViewModel(sortOrder),
            FilterViewModel = new
FilterViewModel(_context.Years.ToList(), year, _context.Regions.ToList(),
region),
            Indicators = items,
```

```
        Info = _context.Info
    };


    return View(viewModel);
}

// GET: Indicators/Details/5
[Authorize(Roles = "admin")]
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var indicatorsModel = await _context.Indicators
        .Include(i => i.Region)
        .Include(i => i.Year)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (indicatorsModel == null)
    {
        return NotFound();
    }

    return View(indicatorsModel);
}

// GET: Indicators/Create
```

```
        [Authorize(Roles = "admin")]
        public IActionResult Create()
        {
            ViewData["RegionName"] = new SelectList(_context.Regions,
"Id", "Name");
            ViewData["YearNumb"] = new SelectList(_context.Years, "Id",
"YearNumb");
            return View();
        }

        // POST: Indicators/Create
        // To protect from overposting attacks, enable the specific properties
you want to bind to, for
        // more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "admin")]
        public async Task<IActionResult>
Create([Bind("Id,YearId,RegionId,Vrp,NumberOfEnterprises,WasteGenera
tion,ExpendituresOnEnvProt,TotalEmissions,CarbonMonoxide,Methane,Ni
trogenDioxide,NitricOxide,Soot,SulfurDioxide,NonMetOrgCompounds,Ca
rbonDioxide,FromMobileSources")] IndicatorsModel indicatorsModel)
        {
            if (ModelState.IsValid)
            {
                _context.Add(indicatorsModel);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
```

```
            ViewData["RegionId"] = new SelectList(_context.Regions, "Id",
"Id", indicatorsModel.RegionId);
            ViewData["YearId"] = new SelectList(_context.Years, "Id", "Id",
indicatorsModel.YearId);

            return View(indicatorsModel);
        }

        // GET: Indicators/Edit/5
        [Authorize(Roles = "admin")]
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var indicatorsModel = await _context.Indicators.FindAsync(id);
            if (indicatorsModel == null)
            {
                return NotFound();
            }
            ViewData["RegionName"] = new SelectList(_context.Regions,
"Id", "Name");
            ViewData["YearNumb"] = new SelectList(_context.Years, "Id",
"YearNumb");
            return View(indicatorsModel);
        }
```

```
        // POST: Indicators/Edit/5
        // To protect from overposting attacks, enable the specific properties
    you want to bind to, for
        // more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "admin")]
        public async Task<IActionResult> Edit(int id,
    [Bind("Id,YearId,RegionId,Vrp,NumberOfEnterprises,WasteGeneration,Ex
    pendituresOnEnvProt,TotalEmissions,CarbonMonoxide,Methane,Nitrogen
    Dioxide,NitricOxide,Soot,SulfurDioxide,NonMetOrgCompounds,CarbonD
    ioxide,FromMobileSources")] IndicatorsModel indicatorsModel)
        {
            if (id != indicatorsModel.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(indicatorsModel);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!IndicatorsModelExists(indicatorsModel.Id))
                    {
```

```
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["RegionId"] = new SelectList(_context.Regions, "Id",
"Id", indicatorsModel.RegionId);
    ViewData["YearId"] = new SelectList(_context.Years, "Id", "Id",
indicatorsModel.YearId);
    return View(indicatorsModel);
}

// GET: Indicators/Delete/5
[Authorize(Roles = "admin")]
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var indicatorsModel = await _context.Indicators
        .Include(i => i.Region)
        .Include(i => i.Year)
        .FirstOrDefaultAsync(m => m.Id == id);
```

```
            if (indicatorsModel == null)
            {
                return NotFound();
            }

            return View(indicatorsModel);
        }

        // POST: Indicators/Delete/5
        [Authorize(Roles = "admin")]
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var indicatorsModel = await _context.Indicators.FindAsync(id);
            _context.Indicators.Remove(indicatorsModel);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool IndicatorsModelExists(int id)
        {
            return _context.Indicators.Any(e => e.Id == id);
        }
    }
}
```

ДОДАТОК 3

Система моніторингу рівня низьковуглецевого розвитку України

Опис програмного модулю

УКР.НТУУ «КПІ ім. І.Сікорського». ТМ4110_18Б 13-1

Аркушів 10

2020

АНОТАЦІЯ

Основний сервіс системи розроблений з використанням мови C# на платформі ASP.NET Core. Сервіс входить до рівня логіки, призначений для обробки вхідних даних та відповідає за обробку отриманих від рівня представлення даних, реалізує всю необхідну логіку додатка, всі обчислення, взаємодіє з рівнем доступу до даних і передає рівню представлення результат обробки.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Даний модуль представляє собою клас, який надає можливість переглядати, створювати, редагувати та видаляти дані.

Для функціонування програми необхідно встановлений Microsoft Visual Studio 2019 з вбудованим IIS Express, встановлені пакети Microsoft.Entity.FrameworcCore (3.1.4), Microsoft.Entity.FrameworcCore. SqlServer (3.1.4). В браузері повинен бути увімкнена підтримка JavaScript та cookie файлів.

Мови програмування, які використовувались при розробці: C# та JavaScript.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний модуль призначений для обміну даними з рівнем доступу до даних та рівнем представлення.

Програмний модуль виконує такі задачі:

- створення, редагування, збереження та видалення даних;
- фільтрація, пагінація та сортування даних
- валідація вхідних даних;
- передача необхідних колекцій у рівень представлення.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Рівень логіки містить набір компонентів, які відповідають за обробку отриманих від рівня представлення даних, реалізує всю необхідну логіку додатка, всі обчислення, взаємодіє з рівнем доступу до даних і передає рівню представлення результат обробки.

Рівень логіки включає в себе об’єкти для передачі даних з шару доступу до даних в шар представлення:

- Index.cshtml
- Create.cshtml
- Delete.cshtml
- Details.cshtml
- Edit.cshtml
- Correlation.cshtml
- InfoSource.cshtml
- RegionStat.cshtml
- YearsStat.cshtml.

Рівень логіки також включає в себе сервіси, які забезпечують зв’язок між шаром представлення та доступу до даних, валідацію даних та виконують усі основні операції в програмі:

1) IndicatorsController– клас, який реалізує містить наступні поля та методи:

- private readonly IndicatorsDbContext _context – екземпляр класу IndicatorsDbContext, через який відбувається зв’язок з рівнем доступу до даних;
- public async Task<IActionResult> Index(int? year, int? region, int page = 1, SortState sortOrder = SortState.YearDesc) – асинхронний метод,

який забезпечує сортування, фільрацію, пагінацію та передачу обробленої інформації у представлення. Має обмеження на використання лише користувача з роллю “admin”

- public async Task<IActionResult> Details(int? id) – асинхронний метод, який повертає інформацію про обраний запис таблиці у представлення. Має обмеження на використання лише користувача з роллю “admin”
- public IActionResult Create() – метод, який повертає представлення для додання нового запису у таблицю. Має обмеження на використання лише користувачей з роллю “admin”
- public async Task<IActionResult> Create([Bind("Id,YearId,RegionId,Vrp,NumberOfEnterprises,WasteGeneration,ExpendituresOnEnvProt,TotalEmissions,CarbonMonoxide,Methane,NitrogenDioxide,NitricOxide,Soot,SulfurDioxide,NonMetOrgCompounds,CarbonDioxide,FromMobileSources")] IndicatorsModel indicatorsModel) – асинхронний метод типу HttpPost, який приймає дані з представлення, проводить валідацію даних, у випадку проходження валідації – додає новий запис до бази даних та зберігає зміни. Має обмеження на використання лише користувачей з роллю “admin”
- public async Task<IActionResult> Edit(int? id) – асинхронний метод, який повертає представлення для редагування обраного запису. Має обмеження на використання лише користувача з роллю “admin”
- public async Task<IActionResult> Edit(int id, [Bind("Id,YearId,RegionId,Vrp,NumberOfEnterprises,WasteGeneration,ExpendituresOnEnvProt,TotalEmissions,CarbonMonoxide,Methane,NitrogenDioxide,NitricOxide,Soot,SulfurDioxide,NonMetOrgCompounds,CarbonDioxide,FromMobileSources")] IndicatorsModel

indicatorsModel) – асинхронний метод типу HttpPost, який приймає дані з представлення, проводить валідацію даних, у випадку проходження валідації – вносить редагування до обраного запису бази даних та зберігає зміни. Має обмеження на використання лише користувача з роллю “admin”

- public async Task<IActionResult> Delete(int? id) – асинхронний метод, який повертає представлення для видалення обраного запису. Має обмеження на використання лише користувача з роллю “admin”
- public async Task<IActionResult> DeleteConfirmed(int id) – асинхронний метод типу HttpPost, який приймає дані з представлення, дає можливість користувачу підтвердити видалення та видаляє обраний запис бази даних та зберігає зміни. Має обмеження на використання лише користувача з роллю “admin”
- private bool IndicatorsModelExists(int id) – приватний метод, який перевіряє наявність запису у таблиці.
- public IndicatorInfoesController(IndicatorsDbContext context) – конструктор за замовчуванням, який приймає контекст даних та передає його у контролер для взаємодії з базою даних.
- public IActionResult Correlation() – метод формування та передачі даних у представлення, в якому відбувається розрахунок матриці кореляції.
- public IActionResult RegionStat(int? region) – метод формування та передачі даних у представлення, в якому відбувається розрахунок та відображення регіональної статистики.
- public IActionResult YearsStat(int? year) – метод формування та передачі даних у представлення, в якому відбувається розрахунок та відображення річної статистики.

public async Task<IActionResult> InfoSource() – метод формування та передачі даних у представлення, для відображення додаткової інформації про індикатори, та посилання на джерела інформації.

-10-

ВХІДНІ ТА ВИХІДНІ ДАНІ

Модуль отримує вхідні дані з шару представлення через параметри методів. Вихідні дані передаються з шару доступу до даних у шар представлення через значення, які повертають методи сервісу.

Методи сервісу умовно поділяються на 2 типи:

– для створення або редагування даних;

– для отримання даних.

Наприклад, метод для створення нового запису: public async Task<IActionResult>Create([Bind("Id,YearId,RegionId,Vrp,NumberOfEnterprises,WasteGeneration,ExpendituresOnEnvProt,TotalEmissions,CarbonMonoxide,Methane,NitrogenDioxide,NitricOxide,Soot,SulfurDioxide,NonMetOrgCompounds,CarbonDioxide,FromMobileSources")] IndicatorsModel indicatorsModel) – вхідні дані: код запису типу int(генерується автоматично), ідентифікатор року типу int, ідентифікатор регіону типу int та значення кожного індикатора типу real. Вхідні дані проходять валідацію та додаються до бази даних.

Вихідні дані методів MonitoringController представленні у вигляді колекцій. Для методу, який повертає дані кореляційного аналізу – обчислена матриця кореляції, для методів які повертають дані статистики – нормалізовані дані, які відповідають запиту користувача. Методи передають дані у представлення, де відбувається візуалізація.

Методи для отримання даних можуть повертати один об’єкт або набір даних (колекції).

ДОДАТОК 4

Система моніторингу рівня низьковуглецевого розвитку України

Апробація

УКР.НТУУ «КПІ ім. І.Сікорського». ТМ4110_18Б

Аркушів 4

2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVIII Міжнародної
науково-практичної конференції
молодих вчених і студентів
2020 року

ТОМ 2

Київ- 2020

УДК 004.094:304.444

Student 4 курсу, гр. ТМ-61 Annenkov M.E.
Assoc.prof., cand.econ.sc. Karaieva N.V.

UKRAINE'S LOW-CARBON DEVELOPMENT MONITORING SYSTEM

On July 18, 2018, at the Cabinet of Ministers meeting, the Government adopted a strategic document on the transition of Ukraine's economy to the low-carbon development model. As outlined in Ukraine's Low-Carbon Development (LCD) Strategy for 2050, low-carbon development is a social and economic development of the country aimed at reducing greenhouse gas emissions. [1]

A multi-parameter system of relevant statistical indicators and indicators is developed and used to measure the level of the LCD, to monitor its dynamics. The creation of a monitoring system, which is a publicly available information resource, should promote democratic transformation in society by ensuring that the public has access to information resources.

The main blocks of the system are shown in (Fig. 1)

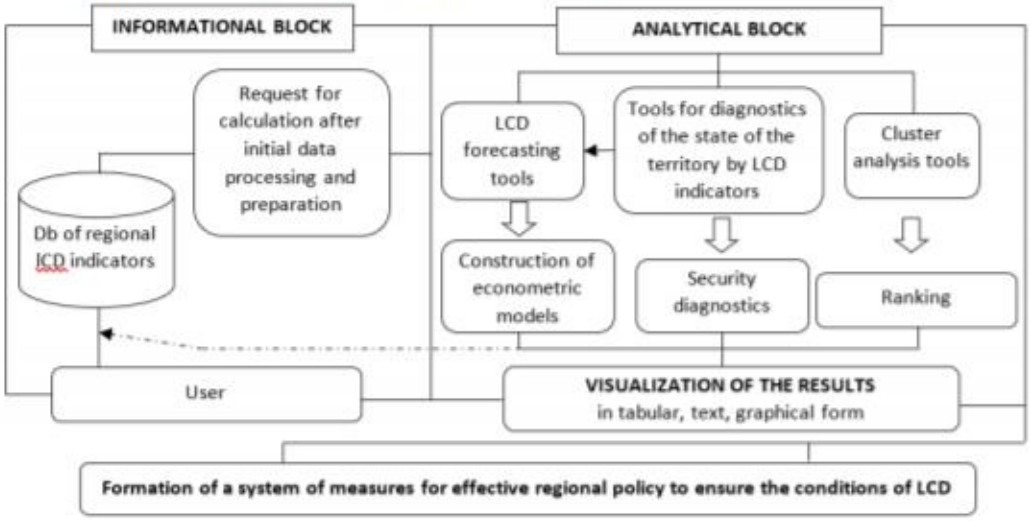


Figure 1. Structure of the system monitoring of LCD indicators

The database (DB) must be represented by object-relational normalized database tables containing a hierarchical system of interrelated metrics that are specified in the LCD strategy. The essence of the database concept is the integrated storage and differentiated use by applications of all information about objects.

The analytics unit should meet the needs of the users to obtain analytical information depending on the type of application tasks. The developed system can be used in planning effective regional policy measures according to the LCD strategy.

As a DB it is proposed to use MS SQL Server in conjunction with Entity Framework technologies. The system is proposed to be implemented as an ASP.NET Core MVC application and deployed on the basis of Amazon Web Services (AWS). The end-user system will be accessible through any web browser.

References:

1. Стратегія низьковуглецевого розвитку України до 2050 року. URL: https://menr.gov.ua/news/31815.html (Date of appeal: 02.26.2020).